

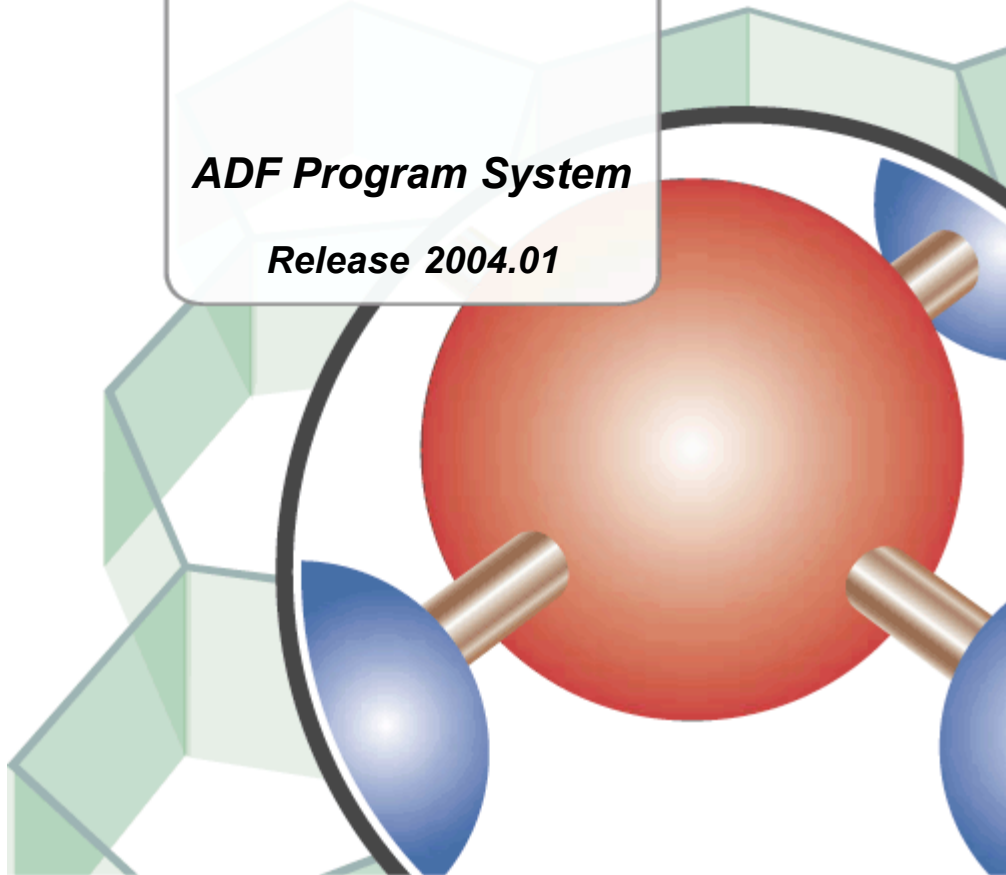


Scientific Computing & Modelling

# ***Examples***

***ADF Program System***

***Release 2004.01***



# Table of Contents

## Examples

### General notes on the Examples

### Sample Runs with ADF

#### Geometry Optimizations

H<sub>2</sub>O: Geometry Optimization

Formaldehyde: another Optimization

AuH: Scalar-Relativistic Optimization

H<sub>2</sub>O: restraint Geometry Optimization

#### ZORA and spin-orbit Relativistic Effects

Au<sub>2</sub>: ZORA Relativistic Effects

Bi and Bi<sub>2</sub>: Spin-Orbit

Tl: Spin-Orbit unrestricted non-collinear

#### IR Frequencies, transition state searches, linear transits, intrinsic reaction coordinates

NH<sub>3</sub>: Frequencies

HCN: LT, Frequencies, TS, and IRC

CH<sub>4</sub>+HgCl<sub>2</sub>⇌CH<sub>3</sub>HgCl+HCl: a TS search

H<sub>2</sub>O: constraint Linear Transit

#### Analysis options: fragment orbitals and bond energy decomposition

Ni(CO)<sub>4</sub>: Compound Fragments

PtCl<sub>4</sub>H<sub>2</sub><sup>2-</sup>: Fragments again

H<sub>2</sub>: Spin-unrestricted Fragments

PCCP: Bond Energy analysis open-shell fragments

TIH: Spin-Orbit SFO analysis

#### Various applications with ADF

Cr(NH<sub>3</sub>)<sub>6</sub>: Multiplet States

Cr(CO)<sub>5</sub>+CO: Basis Set Superposition Error

N<sub>2</sub><sup>+</sup>: Localized Hole

NNO: Core-electron binding energies

HCl: Solvent Effects

N<sub>2</sub> and PtCO: Electric Field, Point Charge(s), use of Basis keyword

CuH<sup>+</sup>: calculation of S<sup>2</sup>

#### Time-dependent DFT applications

Au<sub>2</sub>: Response Properties

Hyperpol: Hyperpolarizabilities of He and H<sub>2</sub>

HF: Dispersion Coefficients

DMO: Circular Dichroism spectrum

DMO: Optical Rotation Dispersion

#### Special exchange-correlation functionals

CO: asymptotically correct xc potentials

OH: Meta-GGA energy functionals

H: SIC-VWN potential

#### QM/MM calculations

QMMM\_Butane: Basic QMMM Illustration

QMMM\_CYT

QMMM\_Surface: Ziegler-Natta catalysis

**POST-ADF Property and utility programs**

NMR chemical shifts and spin-spin coupling constants  
HBr: NMR Chemical Shifts  
HgMeBr: NMR Chemical Shifts  
CH<sub>4</sub>: NMR Chemical Shifts, SAOP potential  
CO: NMR Chemical Shifts, SIC-VWN potential  
PF<sub>3</sub>: NMR Properties, Nucleus-independent chemical shifts  
PF<sub>3</sub>: Comparison of NMR with EPR/NMR  
VOCl<sub>3</sub>: NMR Properties  
C<sub>2</sub>H<sub>2</sub>: Nuclear Spin-spin coupling constants  
ESR / EPR properties  
TiF<sub>3</sub>: ESR Properties  
VO: collinear approximation  
Ge<sup>+</sup> and H<sub>2</sub><sup>+</sup>: EPR spectrum  
Analytic second derivatives  
CN: Analytic second derivatives  
CH<sub>4</sub>: Analytic second derivatives  
HI: ZORA Analytic second derivatives  
Post-ADF analysis utilities  
NO<sub>2</sub>: Contour Plots using *Densf* and *Cntrs*  
C<sub>2</sub>H<sub>2</sub>: Localization of Molecular Orbitals  
Cu<sub>4</sub>CO: Density of States

**Input for third party Software**

adf2aim: convert an ADF TAPE21 to WFN format (for Bader analysis)  
NBO analysis: adfnbo, gennbo

**Sample Runs with BAND**

NaCl: Bulk Crystal  
Cu\_slab: 2-dim. Periodic System  
CO absorption on a Cu slab  
Polyacetylene polymer calculation  
Relativistic effects: Platinum slab  
Pd bulk  
Hydrogen on Pt surface  
NiCu surface alloy with GGA xc potential  
Confinement,tails, and nonorthogonalSCFbasis options for copper dimer  
Time-dependent DFT calculations for bulk silicon  
Time-dependent DFT calculations for hydrogen chain

## General notes on the Examples

The ADF package contains a series of sample runs. Provided are UNIX scripts to run the calculations and the resulting output files. The first part of each output is an echo of the input (for most of the programs in the package).

The examples serve:

- To check that the program has been installed correctly: run the sample inputs and compare the results with the provided outputs. *Read the remarks below about such comparisons.*
- To demonstrate how to do calculations: an illustration to the User manuals. The number of options available in ADF is substantial and the sample runs do not cover all of them. They should be sufficient, however, to get a feeling for how to explore the possibilities.
- To work out special applications that do not fit well in the User's Guide.

Where references are made to the operating system (OS) and to the file system on your computer, the terminology of a UNIX type OS is used and a hierarchical structure of *directories* is assumed.

All sample files are stored in subdirectories under \$ADFHOME/examples/, where \$ADFHOME is the main directory of the ADF package. There are two main subdirectories in examples/: *adf/* for calculations with the molecular code ADF (and related utility programs) and *band/* for calculations with the periodic structures code BAND. Each sample run has its own directory (under *adf/* or *band/* respectively). For instance, \$ADFHOME/examples/adf/e\_HCN/ contains an ADF calculation on the HCN molecule. Each sample subdirectory contains:

- A file run: the UNIX script to execute the calculation or sequence of calculations of the example
- A file output: the resulting output(s) against which you can compare the outcome of your own calculation.

### Notes:

- The UNIX scripts make use of the *rm* (remove) command. Some UNIX users may have aliased the *rm* command. They should accordingly adapt these commands in the sample scripts so as to make sure that the scripts will remove the files.  
New users may get stuck initially because of files that are lingering around after an earlier attempt to run one of the examples. In a subsequent run, when the program tries to open a similar (scratch) file again, an error may occur if such a file already exists. Always make sure that no files are left in the run-directory except those that are required specifically.
- When you run the samples, be aware that they contain rather general *rm* commands. Starting from the examples in adf2004.01 the rather general *rm* commands have been replaced with specific *rm* commands, which makes it less likely that something is removed that you would rather keep. But still take care that nothing is removed that you would rather keep! It is a good idea to use for such tests a safe 'temp' directory that contains no other important stuff. All will be OK if you run them in the directory in which the pertaining sample resides. Be sure to check and understand what will happen if you move and execute them somewhere else.
- The run-scripts use the environment variables ADFBIN and ADFRESOURCES. They stand respectively for the directory that contains the program executables and the main directory of the database. To use the scripts as they are you must have defined the variables ADFBIN and ADFRESOURCES in your environment.  
If a parallel (PVM or MPI) version has been installed, it is preferable to have also the environment variable

nscm. This defines the default number of parallel processes that the program will try to use. In addition, you may want to create a file \$ADFBIN/nodeinfo to specify the maximum number of parallel processes to use for each of the hosts in the virtual parallel machine (hosts not mentioned in this file will have at most one process spawned on them). Consult the Installation Manual for details.

- As you will note the sample run scripts refer to the programs by names like 'adf', 'band', and so on. When you inspect your \$ADFBIN directory, however, you may find that the program executables have names 'adf.exe', 'band.exe'. There are also files in \$ADFBIN with names 'adf', 'band', but these are in fact scripts to execute the binaries. We strongly recommend that you use these scripts in your calculations, in particular when running parallel jobs: the scripts take care of some aspects that you have to do otherwise yourself in each calculation.
- You need a license file to run any calculations successfully. If you have troubles with your license file, consult the Installation manual. If that doesn't help contact us at [support@scm.com](mailto:support@scm.com)

Many of the provided samples have been devised to be short and simple, at the expense of physical or chemical relevance and precision or general quality of results. They serve primarily to illustrate the use of input, necessary files, and type of results. The descriptions have been kept brief. Extensive information about using keywords in input and their implications is given in the User's Guides (ADF and BAND) and the Utilities and Property Programs documents (NMR, DIRAC, and other utility programs).

When you compare your own results with the sample outputs, you should check in particular (as far as applicable):

- Occupation numbers and energies of the one-electron orbitals;
- The optimized geometry;
- Vibrational frequencies;
- The bonding energy and the various terms in which it has been decomposed;
- The dipole moment;
- The logfile. At the end of a calculation the logfile is copied automatically (by the program itself) to the tail of standard output.

#### General remarks about comparisons:

- For technical reasons, the discussion of which is beyond the scope of this document, differences between results obtained on different machines, or with different numbers of parallel processes, may be much larger than you would expect. They may indeed exceed significantly the machine precision. What you should check is that they fall well (by at least an order of magnitude) within the *numerical integration* precision used in the calculation.
- For similar reasons the orientation of the molecule used by the program may be different on different machines, even when the same input is supplied. In such cases the different orientations should be related and only differ in some trivial way, such as by a simple rotation of all coordinates by 90 degrees around the z-axis. When in doubt, contact an ADF representative.
- An ADF run may generate, apart from result files that you may want to save, a few scratch files. The UNIX scripts that run the samples take care of removing these files after the calculations have finished, to avoid that the program aborts in the next run by attempting to open a 'new' file that is found to exist already.
- A sample calculation may use one or more data files, in particular *fragment* files. The samples are self-contained: they first run the necessary pre-calculations to produce the fragment files. In 'normal' research work you may have libraries of fragments available, first for the 'basic atoms', and later, as projects are developing, also for larger fragments so that you can start immediately on the actual system by

attaching the appropriate fragment files.

Default settings of print options result in a considerable amount of output. This is also the case in some of the sample runs, although in many of them quite a bit of 'standard' output is suppressed by inserting applicable print control keys in the input file. Consult the User's Guide about how to regulate input with keys in the input file.

## Survey of the Examples

The Survey of Applications follows a survey of the main application topics with references to related sample runs is given. A sample run usually involves several calculations, for instance a few CREATE runs (with ADF), then a molecular calculation (also ADF), and finally a NMR calculation (with the NMR program) to compute chemical shifts. The samples are identified in this documentation by the name of the directory they reside in. All such names start with 'e\_'. The samples are indicated by these directory names, however with the 'e\_' prefix omitted. For instance, GO\_H2O refers to the directory e\_GO\_H2O/ (in \$ADFHOME/adf/), where in this case GO stands for Geometry Optimization.

# Sample Runs with ADF

## Geometry Optimizations

### H<sub>2</sub>O: Geometry Optimization

*Sample directory:* adf/e\_GO\_H2O/

Geometry optimization of the water molecule, using the (default) *local* density functional approximation (LDA)

Fair quality basis set: triple zeta with polarization. Three equivalent computations are carried out. In all three the atomic coordinates are input in Z-matrix format. The optimization is carried out by optimization of the *internal* coordinates in the first two calculations, and by optimizing the *Cartesian* coordinates in the third one. In calculation #2 the start-up Hessian is defined in the input file; in #1,3 the default start-up Hessian (from a force-field approximation) is applied.

As expected all final results are the same, within the range that might be expected from the convergence thresholds (here: the default values).

The '-n' flag, with value one (1) in the commands \$ADFBIN/adf is used to control parallelization. 'adf' and other program names that you may find in \$ADFBIN are not the executables themselves but (UNIX) scripts to control running the corresponding programs. If a parallel version has been installed and the machine configuration is right, you can carry calculations out in parallel by supplying a suitable value to the -n flag. Omitting the -n flag invokes the default value, which is given by the environment variable \$NSCM. Finally, depending on the type of parallel platform, a file \$ADFBIN/nodeinfo may define an upper bound on the parallel execution. See the Installation manual for details on the parallel installation.

In all subsequent examples, the set-shell and remove-file commands will be omitted, as well as any -n flags. Also any inputs for create runs will not be shown in other examples except when a special feature is involved or when it may help to clarify the example at hand.

### Z-matrix Optimization

```
$ADFBIN/adf <<eor
Title WATER Geometry Optimization with Internal Coordinates
```

```
Atoms  Z-Matrix
 1. O  0 0 0
 2. H  1 0 0  rOH
 3. H  1 2 0  rOH theta
End
```

```
Basis
Type TZP
Core Small
End
```

```
GeoVar
rOH=0.9
```

```
theta=100
End
```

```
Geometry
End
```

```
End Input
eor
```

A title is supplied. This title is printed in the output header. It is also written to any result files from the calculation and will be printed out when such a file is attached to another calculation, for instance as a fragment file. In addition, adf constructs a 'jobidentification' string that contains the adf release number and the date and time. The jobidentification is also printed in the output header and dumped on any result files.

The atomic positions are given with the key atoms. Bond lengths are in Angstrom and angles in degrees. The key geometry assigns numerical starting values to the two variables. We could also have written numerical values directly in the atoms block. The structure used here implies that the two HO bonds are equal and must remain equal: they are associated with the same variable; this constraint would not have applied if numerical data had been put in the atoms section, although the symmetry would have kept them equal anyway.

The key geometry *must* be supplied to let the program do an optimization: otherwise a single point calculation would be carried out. The geometry data block is empty here, meaning that no default values are reset for the options that are controlled with this key.

No symmetry is specified by a Schönflies type symbol (key symmetry). The program will use the true symmetry of the nuclear frame (accounting for any fields, if present). In this case that is C(2v). If such symmetry would not be acceptable for adf (not all pointgroups are supported!) or when you want to run in a lower symmetry, the symmetry to be used must be specified.

The fragment files are defined implicitly with the Bais keyword. In this case (as well as in most other samples) the fragment files reside in the local directory since they were created there in the same job. If they would have been located elsewhere you could specify a full path for each of the files, or alternatively (if all fragmentfiles are in one single directory) write the directory after the keyword fragments (on the same line).

The precision of numerical integration, to evaluate Hamiltonian matrix elements etc., is not specified and attains therefore the default value (4.0 in an optimization run).

## Definition of (diagonal) start-up Hessian

```
$ADFBIN/adf <<eor
Title WATER optimization with (partial) specification of Hessian
```

```
Atoms Z-Matrix
1. O 0 0 0
2. H 1 0 0 rOH
3. H 1 2 0 rOH theta
End
```

```
GeoVar
rOH=0.9
theta=100
End
```

```
HessDiag rad=1.0 ang=0.1
```

```
Fragments  
H t21.H  
O t21.O  
End
```

```
Geometry  
End
```

```
End Input  
eor
```

All input is identical to the previous case, except for the key HessDiag. This defines here the start-up Hessian to be diagonal with values 1.0 and 0.1 for the entries related to bondlengths and angles respectively.

## Optimization in Cartesian coordinates

```
$ADFBIN/adf <<eor  
Title WATER Geometry Optimization in Cartesians
```

```
Geometry  
  Optim Cartesian  
End
```

```
Define  
rOH=0.9  
theta=100  
End
```

```
Atoms  Z-Matrix  
1. O  0 0 0  
2. H  1 0 0  rOH  
3. H  1 2 0  rOH theta  
End
```

```
Fragments  
H t21.H  
O t21.O  
End
```

```
End Input  
eor
```

In the last calculation the atomic coordinates are input in the same way as before, but the geometry block now specifies, with the subkey *optim*, that the *cartesian* coordinates are to be varied and monitored for convergence.

If different coordinates are specified in the *optim* instruction than were used for the input in the atoms block, no constraints can be used. The variable 'rOH' cannot be placed in the geovar block therefore, since that would imply a constraint: keep the two OH distances equal.

The placement of rOH (and theta) in the define block has a completely different meaning. define merely

associates a numerical value with an identifier. Wherever the identifier occurs in input (not only in the `atoms` block) it will be replaced by the numerical value. This means that there are now nine (9) variables: the  $x,y,z$  coordinates of the three atoms.

Pure translations and rotations will be filtered out by the program and the symmetry (explicitly specified or internally computed),  $C(2v)$  here, will be enforced on all developments so that the situation is equivalent to the previous calculation as regards the degrees of freedom of the system.

Remark: the `define` block must occur in input *before* the variables defined in it are used. This is one of the few cases where the relative position of keys in the input stream is relevant. The same does not hold for the `geovar` key used in the earlier example: `geovar` may be placed anywhere in the input, irrespective of the locations of atoms.

## Formaldehyde: another Optimization

*Sample directory:* adf/e\_GO\_Formaldehyde/

In the input for the optimization run the atomic coordinates are in Z-matrix format while the optimization variables are the *Cartesian* coordinates. This is achieved with the *optim* subkey in the geometry block.

A single geovar variable is used for different coordinates. However, since the type of optimization variables (Cartesian) is not the same as the type of input coordinates (Z-matrix), no constraints are implied by this. In fact, the related coordinates do remain equal, but this is because they are symmetry related and the program preserves symmetry anyway.

NonLocal gradient corrections (gga: Generalized Gradient Approximation) according to the approach known as 'Becke' (for exchange) and 'Perdew' (correlation) are included self-consistently with the key xc.

```
$ADFBIN/adf << eor
Title formaldehyde

Geometry
  Optim cartes
End

XC
  GGA Becke Perdew
END

Symmetry C(2v)

Atoms Z-matrix
  1 O  0 0 0 0.0 0.0 0.0
  2 C  1 0 0 r2  0.0 0.0
  3 H  2 1 0 r3  a3  0.0
  4 H  2 1 3 r3  a3  t4
End

Fragments
  C t21.C
  O t21.O
  H t21.H
End

Geovar
  r2  1.94
  r3  0.95
  a3  120
  t4 -180
End

integration 4.5

End Input
eor
```

## AuH: Scalar-Relativistic Optimization

*Sample directory:* adf/e\_RelGO\_AuH/

A simple geometry optimization using the scalar relativistic option, implying that relativistic core potentials must be generated first (*dirac*).

```
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/H
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/Au.4d
```

```
mv TAPE12 t12.rel
```

The optimization run is now straightforward (after having created the relativistic basic atoms, not shown here).

```
$ADFBIN/adf << eor
title AuH relativistic optimization
```

```
integration 5.5
```

```
atoms Zmat
Au 0 0 0
H 1 0 0 1.5
end
```

```
fragments
Au t21.Au
H t21.H
end
```

```
xc
GGA Becke Perdew
end
```

```
relativistic
CorePotentials t12.rel &
H 1
Au 2
end
```

```
geometry
convergence grad=1e-4
end
```

```
end input
eor
```

The key COREPOTENTIALS is used as block key *and* it has an argument ('t12.rel'). Consequently the continuation character (&) is used. Note that the *order* of DIRAC runs, to create the relativistic corepotentials file TAPE12, determines that in the key block to the CorePotentials key, the H atom must relate to the first section on TAPE12 and the Gold atom to the second section.

## H<sub>2</sub>O: restraint Geometry Optimization

*Sample directory:* adf/e\_GO\_restraint/

The restraint does not have to be satisfied at the start of the geometry optimization. An extra force is added to restrain the bond length, angle, or diherdral angle to a certain value.

Example for angle restraint

```
$ADFBIN/adf << eor
title WATER geometry optimization with angle restraint
```

```
ATOMS
  1.O      0.001356  0.000999  0.000000
  2.H      0.994442 -0.037855  0.000000
  3.H     -0.298554  0.948531  0.000000
END
```

```
BASIS
Type DZP
END
```

```
INTEGRATION 4 4
```

```
RESTRAINT
ANGLE 3 1 2 125.0
END
```

```
GEOMETRY
END
```

```
endinput
eor
```

Example for bond length restraint

```
$ADFBIN/adf << eor
title WATER Geometry Optimization with bond length restraint
```

```
ATOMS
  1.O      0.001356  0.000999  0.000000
  2.H      0.994442 -0.037855  0.000000
  3.H     -0.298554  0.948531  0.000000
END
```

```
BASIS
Type DZP
END
```

```
INTEGRATION 4 4
```

```
RESTRAINT
```

```
DIST 1 2 1.03
DIST 1 3 1.03
END
```

```
GEOMETRY
END
```

```
endinput
eor
```

Example for dihedral angle restraint

```
$ADFBIN/adf << eor
Title Restraining dihedral of ethane
```

```
SYMMETRY NOSYM
```

```
ATOMS
```

1.C	-0.004115	-0.000021	0.000023
2.C	1.535711	0.000022	0.000008
3.H	-0.399693	1.027812	-0.000082
4.H	-0.399745	-0.513934	0.890139
5.H	-0.399612	-0.513952	-0.890156
6.H	1.931188	0.514066	0.890140
7.H	1.931432	0.513819	-0.890121
8.H	1.931281	-1.027824	0.000244

```
END
```

```
INTEGRATION 4 4
```

```
RESTRAINT
```

```
DIHED 6 2 1 3 20.00
END
```

```
BASIS
```

```
type DZP
END
```

```
GEOMETRY
END
```

```
endinput
eor
```

# ZORA and spin-orbit Relativistic Effects

## Au<sub>2</sub>: ZORA Relativistic Effects

*Sample directory:* adf/e\_Au2\_ZORA/

Another relativistic geometry optimization, now with the ZORA formalism. The build-up is quite similar to the RelGO\_AuH case: DIRAC calculations for the involved atoms to get relativistic core potentials, Create runs and finally the molecular optimization run. In between the Create runs and the molecular optimization run, a single-atom Spin-Orbit calculation is carried out. The Spin-Orbit corrections are not available in optimization calculations, so in the final molecular run, the *scalar* (ZORA) relativistic terms are used.

```
$ADFBIN/adf << eor
```

```
Title Au relativistic spinorbit
```

```
Integration 6.5
```

```
Atoms
```

```
Au 0 0 0
```

```
End
```

```
Fragments
```

```
Au t21.Au
```

```
End
```

```
XC
```

```
GGA Becke Perdew
```

```
End
```

```
Relativistic SpinOrbit ZORA
```

```
Corepotentials t12.rel
```

```
end input
```

```
eor
```

Since only one type of atom is used, the CorePotentials key can be used as simple key: the data block is not necessary since the program takes (by default) the first section on the TAPE12 file for the first (here: only) atom type in the calculation.

```
$ADFBIN/adf << eor
```

```
Title Au2 relativistic optimization: scalar ZORA
```

```
Integration 6.5
```

```
Atoms Zmat
```

```
Au 0 0 0
```

```
Au 1 0 0 2.5
```

```
End
```

```
Fragments
  Au t21.Au
End

XC
  GGA Becke Perdew
End

Relativistic scalar ZORA
CorePotentials t12.rel

Geometry
  convergence grad=1e-4
End

End Input
eor
```

## Bi and Bi<sub>2</sub>: Spin-Orbit

*Sample directory:* `adf/e_SO_Bi2/`

Application of the Spin-Orbit relativistic option (using double-group symmetry) to Bismuth (atom and dimer).

To prepare for the relativistic calculations, the *dirac* program is applied to generate the relativistic core potential for the Bismuth atom with a frozen core up to the 5p shell.

```
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/Bi.5p
```

```
mv TAPE12 t12rel
```

The next step is the creation of the restricted Bismuth atom (scalar relativistic).

The GGA (Becke-Perdew) facility is used for consistency with the calculations to follow, but is not necessary *per se* to carry out the subsequent calculations.

```
$ADFBIN/adf <<eor
create Bi file=$ADFRESOURCES/TZP/Bi.5p
xc
  LDA vwn
  GGA becke perdew
end
relativistic scalar
corepotentials t12rel &
Bi 1
end
end input
eor
```

```
mv TAPE21 t21Bi
```

Note that usage of the *block* form for the CorePotentials key would not have been necessary here. We could as well have used:

```
corepotentials t12rel
```

instead of

```
corepotentials t12rel &
Bi 1
end
```

### Bi: single atom

For comparison with the full double-group calculation, the 'standard' unrestricted calculation on Bismuth is carried out, using the *scalar* relativistic option.

A net spin polarization of 3 electrons is applied (key charge).

```
$ADFBIN/adf <<eor
```

```

title Bi unrestricted

integration 4.0

xc
  LDA vwn
  GGA becke perdew
end

relativistic scalar
corepotentials t12rel &
Bi 1
end

ATOMS
Bi 0.000000 0.000000 0.00000000
end

fragments
Bi t21Bi
end

unrestricted

charge 0 3

end input
eor

```

The CHARGE key, in conjunction with the UNRESTRICTED key is used to specify that 3 electrons must be unpaired (second value of the CHARGE key), while the system is neutral (first value of the CHARGE key).

Next we do a Spin-Orbit calculation on the Bismuth atom.

Note that it is a 'restricted' run (the key unrestricted is not used). The double-group symmetry orbitals are, like the single-group ones in a non-SpinOrbit calculation, degenerate, allowing 2 electrons in each spatial orbital. These are equally occupied (using fractional occupations if necessary) and the electronic charge density is not spin-polarized.

```

$ADFBIN/adf <<eor
title Bi spinorbit

integration 4.0

xc
  LDA vwn
  GGA becke perdew
end

relativistic spinorbit
corepotentials t12rel &
Bi 1
end

ATOMS

```

```
Bi 0.000000 0.000000 0.00000000
end

fragments
Bi t21Bi
end

end input
eor
```

Comparison of the bonding energy (w.r.t. the create restricted atom) for the scalar relativistic and spin-orbit runs respectively show that application of the spin-orbit operator lowers the energy by approximately 1.1 eV.

In the previous run default occupations were used: the occupations were determined from the aufbau principle during the first few scf iterations.

The following is an excited state calculation: occupation numbers are specified in input and by comparison with the result from the previous run we see that one electron has been promoted from a  $p_{1/2}$  to a  $p_{3/2}$  orbital.

```
$ADFBIN/adf <<eor
```

```
title Bi spinorbit, specified occupations
```

```
PRINT SpinOrbit
```

```
integration 4.0
```

```
xc
LDA vwn
GGA becke perdew
end
```

```
relativistic spinorbit
corepotentials t12rel &
Bi 1
end
```

```
ATOMS
Bi 0.000000 0.000000 0.00000000
end
```

```
fragments
Bi t21Bi
end
```

```
charge 0
```

```
occupations
s1/2 2
p1/2 1
p3/2 2
d3/2 4
d5/2 6
```

```
end
```

```
end input
eor
```

The PRINT key (here with argument SPINORBIT) controls output printing. Here it induces the printing of some extra information about the relativistic double group symmetry orbitals.

## Bi<sub>2</sub> dimer

Now we turn to the dimer Bi<sub>2</sub>: a series of Single Point calculations, all with the same inter atomic distance.

First the scalar relativistic run.

```
$ADFBIN/adf <<eor
title Bi2, scalar relativistic

integration 4.0

relativistic scalar
corepotentials t12rel &
Bi 1
end

ATOMS
Bi 0.0 0.0 1.33
Bi 0.0 0.0 -1.33
end

fragments
Bi t21Bi
end

xc
LDA vwn
GGA becke perdew
end

end input
eor
```

```
mv tape21 t21Bi2
```

The result file tape21 is used as reference in subsequent calculations: run the spin-orbit case starting from the just completed dimer calculation as a fragment. The resulting 'bonding energy', ie the energy w.r.t. the scalar relativistic dimer, gives directly the effect of the full-relativistic versus the scalar relativistic option: the energy is lowered by 2.3 eV.

```
$ADFBIN/adf <<eor
title Bi2 from fragment Bi2, with SpinOrbit coupling

PRINT SpinOrbit

integration 4.0
```

```
relativistic spinorbit
corepotentials t12rel &
Bi 1
end
```

```
ATOMS
Bi 0.0 0.0 1.33 f=Bi2
Bi 0.0 0.0 -1.33 f=Bi2
end
```

```
fragments
Bi2 t21Bi2
end
```

```
xc
LDA vwn
GGA becke perdew
end
```

```
end input
eor
```

```
rm TAPE21 logfile
```

A final consistency check: run the spin-orbit dimer from single-atom fragments. The bonding energy should equal the sum of the bonding energies of the previous two runs: scalar relativistic dimer w.r.t. single atom fragments plus spin-orbit dimer w.r.t. the scalar relativistic dimer.

```
$ADFBIN/adf <<eor
title Bi2 from atomic fragments, SpinOrbit coupling
```

```
PRINT SpinOrbit
```

```
integration 4.0
```

```
relativistic spinorbit
corepotentials t12rel &
Bi 1
end
```

```
ATOMS
Bi 0.0 0.0 1.33
Bi 0.0 0.0 -1.33
end
```

```
fragments
Bi t21Bi
end
```

```
xc
LDA vwn
GGA becke perdew
end
```

end input  
eor

## TI: Spin-Orbit unrestricted non-collinear

*Sample directory:* adf/e\_TI\_noncollinear/

Application of the Spin-Orbit relativistic option (using double-group symmetry, in this case NOSYM) to TI using the collinear and non-collinear approximation for unrestricted Spin-Orbit calculations

Note: For the collinear and the non-collinear approximation one should use symmetry NOSYM and use the key UNRESTRICTED.

The non-collinear example:

```
$ADFBIN/adf << eor
Title TI spinorbit noncollinear
Atoms
  TI 0 0 0
End

Relativistic Spinorbit ZORA
COREPOTENTIALS t12.rel &
  TI 1
End

XC
  gradients becke perdew
end

symmetry nosym
unrestricted
noncollinear

Fragments
  TI t21.TI
End

End input
eor
```

If one replaces the key NONCOLLINEAR with COLLINEAR the collinear approximation will be used instead of the non-collinear approximation. In the case of the collinear approximation default the direction of the magnetization is in the direction of the z-axis. In the non-collinear approximation the magnetization can differ in each point in space.

## IR Frequencies, transition state searches, linear transits, intrinsic reaction coordinates

## NH<sub>3</sub>: Frequencies

*Sample directory:* adf/e\_Freq\_NH3/

Computation of frequencies by Cartesian displacements. The assumed equilibrium input structure is given in internal coordinates. A dummy atom is used for a convenient definition of the Z-matrix such that it reflects the pointgroup symmetry C(3v).

```
$ADFBIN/adf <<eor
title NH3 frequencies

define
  rNH=1.02
  theta=112
  phi=120
end

atoms Z-matrix
  XX  0 0 0
  N   1 0 0 1.0
  H   2 1 0 rNH theta
  H   2 1 3 rNH theta phi
  H   2 1 4 rNH theta phi
end

Basis
Type TZP
Core Small
End

geometry
  optim cartesian
  frequencies
end

thermo T=300,400

integration 5.0

end input
eor
```

The symmetry is determined automatically by the program as C(3v), from the input coordinates. In a Frequencies calculation the symmetry (specified on input or computed internally) is used for analysis and in some cases to speed up the calculation.

The equilibrium coordinate values are supplied as identifiers that are associated with values in the define block.

Unlike using the geovar key, applying the define key does not mean anything in the sense that the various coordinates that refer to the same identifier would be forced to remain equal; it is just a way to display (to the human reader) symmetry in the equilibrium values, to avoid typing errors and to allow an easy adjustment

of starting coordinates for another calculation.

Since the atomic coordinates are input in Z-matrix format, the program would by default carry out displacements in internal coordinates to scan the energy surface and hence compute force constants and frequencies. This is overridden by specifying in the geometry block `optim cartesian`: carry out *cartesian* displacements.

The key `thermo` addresses the thermodynamical analysis (only available in a Frequencies calculation, otherwise ignored). The specification `'T=300,400'` means that the thermodynamic properties are printed for the temperature range 300-400K, in steps of 10K (default) and for a pressure of 1.0 atmosphere (default).

Frequencies calculations suffer easily from numerical inaccuracies. Therefore, the default numerical integration precision in a Frequencies calculation is much higher than in an ordinary single-point or minimization run. Here we specify the `INTEGRATION` level to be 5.0 (quite high, but the default for Frequencies is even 6.0).

## HCN: LT, Frequencies, TS, and IRC

Sample directory: `adf/e_HCN/`

### Summary

- For a sequence of intermediates, each defined by a fixed angle H-C-N between the linear extremes HCN and CNH, the remaining geometrical parameters are optimized, giving a Linear Transit point-by-point scan of the energy curve of the Hydrogen atom travelling from one end of the CN fragment to the other. This is a useful way to get a reasonable first guess of the Transition State.
- At the approximate TS a Frequencies calculation is performed to obtain a fairly accurate Hessian for the next calculation.
- A TS search is carried out, using the computed Hessian. As variation, the TS search is repeated, first with the automatic (internal) Hessian (based on force fields) and then also with a constraint applied.
- A full IRC scan of the full path, starting from the TS, down to the two minima.

### LT

The first calculation is a Linear Transit where the Hydrogen atom moves from one side of CN to the other by a parameterized step-by-step change of the angle H-C-N. The other coordinates of the system are optimized along the path.

In the atoms block, one coordinate value is represented by an identifier (th). In the geovar block this is assigned two values, implying that it is a Linear Transit parameter. The initial and final values for the parameter are given.

Since the geometry block does not have OPTIM SELECTED, all other coordinates are optimized for each of the 10 Linear Transit points.

The subkey *iterations* in the geometry block carries *two* arguments: the first is the maximum number of optimization steps (per LT point). The second is the number of LT points to compute in this run: 4. This implies that only a part of the 10-point path defined by the LT parameter(s) will be scanned. The remainder will be done in a follow-up run to illustrate usage of the restart facility.

```
$ADFBIN/adf <<eor
Title   HCN Linear Transit, first part
NoPrint SFO, Frag, Functions, Computation
```

```
Atoms   Internal
  1 C 0 0 0    0 0 0
  2 N 1 0 0    1.3 0 0
  3 H 1 2 0    1.0 th 0
End
```

```
Basis
Type DZP
End
```

```
Symmetry NOSYM
```

Integration 6.0 6.0

Geometry

```
LinearTransit 10
Iterations 30 4
Converge Grad=3e-2, Rad=3e-2, Angle=2
END
```

Geovar

```
th 180 0
End
```

End Input

eor

```
mv TAPE21 t21.LT
rm logfile
```

The NoPRINT key turns off a lot of default output. There are several PRINT and NOPRINT options; see the User's Guides for details.

Since the geometry changes from linear to planar (and finally back to linear again), the symmetry must be given explicitly in the input file. Otherwise the program would find a C(lin) symmetry for the initial geometry and assume that this symmetry is preserved throughout. This would of course result in an error abort when the first LT step is carried out, breaking the linear symmetry.

The here specified symmetry (NOSYM: no symmetry at all) is not the true symmetry of the complete path – C(s) – but a subgroup. It is always allowed to specify a lower symmetry than the actually present symmetry. Such may be necessary (for instance when the true symmetry cannot be handled by adf) or in special cases required for reasons of analysis. Generally speaking, however, we recommend to use the highest symmetry possible (given the case at hand and taking into account the symmetries recognizable by ADF) to boost performance.

Convergence thresholds in the geometry block are set less tight than the defaults: we need only a reasonable estimate of the path, but no highly converged geometries.

At the end of the run the tape21 result file is saved and renamed *t21.LT* to serve as restart file for the follow-up calculation.

## LT continuation

```
$ADFBIN/adf <<eor
Title HCN Linear Transit
NoPrint SFO,Frag,Functions,Computation
```

```
Restart t21.LT
```

Fragments

```
N t21.N
C t21.C
H t21.H
End
```

```
Atoms Internal
1 C 0 0 0 0 0 0
```

```

2 N 1 0 0    1.3 0 0
3 H 1 2 0    1.0 th 0
End

```

```

symmetry NOSYM

```

```

Integration 6.0 6.0

```

```

Geometry
  LinearTransit 10
  Converge      Grad=3e-2, Rad=3e-2, Angle=2
END

```

```

Geovar
  th 180 0
End

```

```

End Input
eor

```

```

rm TAPE21 logfile

```

From the restart file, supplied with the key restart, the program reads off that the first 4 points of the LT path have been done already and the scan is continued with It point #5. The same path definition is supplied again, including the *original* starting values for the coordinates. The actual starting coordinates (for It point #5) are read from the restart file. The input values, however, serve to define and verify consistency of the defined It path and must therefore be supplied correctly.

The key noprint is used to suppress major parts of standard output: all information pertaining to the sfo analysis, all build-from-fragments information, and the lists of elementary functions in the basis sets and fit sets.

## Frequencies at the estimated Transition State

From the results of the Linear Transit run we can sketch the energy barrier that H passes over when going from one side of the molecule to the other. This yields a reasonable guess for the Transition State.

To check that the so-obtained estimate is adequate we compute the frequencies in that geometry: one of them should be imaginary.

Apart from serving as a check that the TS estimate is not too bad, the computed Hessian will also serve in the follow-up calculation to obtain the true TS.

```

$ADFBIN/adf <<eor
Title  HCN Frequencies in LT max (approx), moderate precision
NoPrint  SFO,Frag,Functions,Computation

```

```

Integration 6.0 6.0

```

```

Fragments
  N t21.N
  C t21.C
  H t21.H
End

```

```
Atoms      Internal
  1 C 0 0 0    0    0  0
  2 N 1 0 0    1.186 0  0
  3 H 1 2 0    1.223 70  0
End
```

```
Geometry
  Frequencies
End
```

```
End Input
eor
```

```
mv TAPE21 t21.Freq
```

Inspection of the output file shows that one of the frequencies is imaginary, as expected (printed as negative), signalling the proximity of the Transition State.

The TAPE21 result file of the calculation is renamed and saved. Later we will use it as a 'restart' file for a ts search, namely to supply the computed Hessian as the initial 'guess' of the Hessian in the (ts) optimization run.

## TS search

Now carry out the Transition State search, starting from the It-derived guess.

In this first attempt to find the TS, no use is made of the tape21 result file from the Frequencies run. That will be done in the next calculation.

```
$ADFBIN/adf <<eor
Title   HCN Transition State, automatic initial Hessian
NoPrint SFO,Frag,Functions,Computation
```

```
Integration 6.0 6.0
```

```
Atoms      Internal
  1 C 0 0 0    0    0  0
  2 N 1 0 0    1.186 0  0
  3 H 1 2 0    1.223 70  0
End
```

```
Fragments
  N t21.N
  C t21.C
  H t21.H
End
```

```
Geometry
  TransitionState
End
```

```
End Input
eor
```

```
rm TAPE21 logfile
```

The TS-search run type is specified in the geometryblock.

No symmetry is specified; the program determines the symmetry to be C(s) and consequently carries out the ts search in that symmetry.

## TS search, using the Hessian from the Frequencies run

```
$ADFBIN/adf <<eor
```

```
Title   HCN Transition State,  initial Hessian from Freq run
```

```
NoPrint SFO,Frag,Functions,Computation
```

```
Restart t21.Freq
```

```
Save   TAPE13
```

```
Integration 6.0 6.0
```

```
Atoms      Internal
```

```
  1 C  0 0 0    0    0  0
```

```
  2 N  1 0 0    1.186  0  0
```

```
  3 H  1 2 0    1.223  70  0
```

```
End
```

```
Fragments
```

```
  N t21.N
```

```
  C t21.C
```

```
  H t21.H
```

```
End
```

```
Geometry
```

```
  TransitionState
```

```
End
```

```
End Input
```

```
eor
```

```
mv TAPE13 t13.TS
```

```
rm TAPE21 logfile
```

The CheckPoint file TAPE13, at normal termination automatically deleted by the program, is here saved, using the SAVE key. TAPE13 is as good a restart file as TAPE21 is, but it is a lot smaller. TAPE21 contains a large amount of information for analysis purposes, while TAPE13 contains essentially *only* restart-type data.

The input is identical to the previous one, except for the restart file. This is used here to provide the Hessian computed in the Frequencies run as the start-up Hessian for the ts optimization. At the same time the atomic coordinates are read off from the restart file and override the values in the input file. This latter aspect could have been suppressed; see the User's Guide for using the restart key.

## Constrained TS search

Finally the ts search where one coordinate is kept frozen, to illustrate a constrained optimization.

```
$ADFBIN/adf <<eor
Title  HCN  constrained TS search
NoPrint SFO,Frag,Functions,Computation
```

```
Restart t21.Freq
```

```
Integration 6.0 6.0
```

```
Atoms  Internal
  1 C 0 0 0  0  0  0
  2 N 1 0 0  rNC  0  0
  3 H 1 2 0  1.223 70  0
End
```

```
GeoVar
  rNC=1.186 F
End
```

```
Fragments
  N t21.N
  C t21.C
  H t21.H
End
```

```
Geometry
  TransitionState
End
```

```
End Input
eor
```

```
rm TAPE21 logfile
rm t21.Freq
```

The geovar key specifies that the nc distance, rNC has the initial value 1.15 and remains frozen ('F').

The fact that the optimization is now carried out in a different subspace of atomic coordinates does not prevent us from using the *t21.Freq* restart file to supply the initial Hessian.

## IRC scan of the reaction path

The IRC calculation is split in three steps, to illustrate the Restart facility applied to the IRC functionality.

In the first only a few points are computed, along one of the two paths leading from the TS to the adjacent minima. Since no explicit directives are given in the input to specify the *direction* of the first path, the so-called 'forward' path is taken. The definition of which is 'forward' and which is 'backward' is in fact quite arbitrary and is determined by the program. See the User's Guide for details.

The saved TAPE13 file from one of the TS calculations is used as restart file. This provides (a) the optimized coordinates of the TS as starting point, (b) the initial Hessian to guide the point-by-point optimizations along the IRC path, and (c) the eigenvector of the lowest Hessian eigenvalue to define the initial direction of the IRC path.

The TAPE13 file from this partial IRC scan is saved to serve as start-up file for the next calculations, which will continue the IRC scan.

In the Geometry key block, the run type is set to IRC and the 'Points' option is used to limit the number of IRC points to compute.

```
$ADFBIN/adf << eor
Title  HCN  IRC partial path (forward)
NoPrint SFO,Frag,Functions, Computation
```

```
Integration 6.0 6.0
```

```
Restart  t13.TS
Save    TAPE13
```

```
Atoms      Internal
  1 C 0 0 0    0    0  0
  2 N 1 0 0    1.186 0  0
  3 H 1 2 0    1.223 70  0
End
```

```
Fragments
```

```
  N  t21.N
  C  t21.C
  H  t21.H
End
```

```
Geometry
```

```
  IRC Points=5
End
```

```
End Input
eor
```

```
mv TAPE13 t13.IRC_1
rm TAPE21 logfile
```

The IRC is continued in the next calculation, using the TAPE13 file from the previous one as restart file. From this file, the program reads the IRC path information computed sofar. By default, it would continue on the 'forward' path, since that was not yet finished. However, in the Geometry key block, we now specify not only that a limited number of points is to be computed in this run (5 again), but we instruct the program also to compute only points on the 'backward' path.

```
$ADFBIN/adf << eor
Title  HCN  IRC partial part (backward)
NoPrint SFO,Frag,Functions, Computation
```

```
Restart  t13.IRC_1
Save    TAPE13
```

```
Integration 6.0 6.0
```

```
Atoms      Internal
  1 C 0 0 0    0    0  0
  2 N 1 0 0    1.186 0  0
  3 H 1 2 0    1.223 70  0
```

END

Fragments

N t21.N

C t21.C

H t21.H

End

Geometry

IRC Points=5 Backward

End

End Input

eor

mv TAPE13 t13.IRC\_2

rm TAPE21 logfile

In the third IRC run, the IRC scan is finished. We start with the TAPE13 file from the previous run and set a maximum of 50 IRC points to compute (which turns out to be sufficient for the complete IRC scan). The program starts on the forward path, continuing where the first (not the previous) had stopped – after 5 points –, completes the forward path, and then continues on the backward path, starting where the second IRC run had stopped. Both paths are finished and a summary of the path characteristics is printed in the final part of the output.

\$ADFBIN/adf << eor

Title HCN IRC completion

NoPrint SFO,Frag,Functions, Computation

Restart t13.IRC\_2

Integration 6.0 6.0

Atoms	Internal
1 C 0 0 0	0 0 0
2 N 1 0 0	1.186 0 0
3 H 1 2 0	1.223 70 0

End

Fragments

N t21.N

C t21.C

H t21.H

End

Geometry

IRC Points=50

End

End Input

eor

## $\text{CH}_4 + \text{HgCl}_2 \rightleftharpoons \text{CH}_3\text{HgCl} + \text{HCl}$ : a TS search

*Sample directory:* adf/e\_ReITS\_CH4\_HgCl2/

A Transition State calculation, including scalar relativistic terms in the Hamiltonian.

First the relativistic core potentials are generated.

```
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/Hg.4d
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/Cl.2p
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/C.1s
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/H
```

```
mv TAPE12 t12.rel
```

Then the (relativistic) Create runs.

```
$ADFBIN/adf << eor
create Hg $ADFRESOURCES/TZP/Hg.4d
relativistic
corepotentials t12.rel &
Hg 1
end
end input
eor
```

```
mv TAPE21 t21.Hg
```

```
$ADFBIN/adf << eor
create Cl $ADFRESOURCES/TZP/Cl.2p
relativistic
corepotentials t12.rel &
Cl 2
end
end input
eor
```

```
mv TAPE21 t21.Cl
```

```
$ADFBIN/adf << eor
create C $ADFRESOURCES/TZP/C.1s
relativistic
corepotentials t12.rel &
C 3
end
end input
eor
```

```
mv TAPE21 t21.C
```

```

$ADFBIN/adf << eor
create H $ADFRESOURCES/TZP/H
relativistic
corepotentials t12.rel &
H 4
end
end input
eor

```

```
mv TAPE21 t21.H
```

In the first Create run (Hg) the CorePotentials key could have been used in its simple form, but in the second (and third and fourth, omitted here) the block form is required to identify the appropriate section on TAPE12 for the atom at hand. In the first case we could have relied on the default: the first section on TAPE12 for the first (in this case: only) atom type.

Note that even for H, which obviously has no frozen core at all, we specify the TAPE12 corepotentials file and indicate the appropriate section for H. The reason is that TAPE12 contains not only the (frozen) *core*, but also the *total* atomic (relativistic) potential.

Finally the TS run.

```

$ADFBIN/adf << eor
TITLE Transition State: CH4 + HgCl2 <====> CH3HgCl + HCl

```

```

noprnt sfo,frag
print atdist

```

```

GEOMETRY
  TransitionState
END

```

```

relativistic
corepotentials t12.rel &
C 3
Hg 1
Cl 2
H 4
end

```

```

XC
Ida VWN Stoll
END

```

```

DEFINE
rHg = 2.30
rX1 = 2.35
rX2 = 2.90
rH1 = 1.10
rH2 = 1.10
rH3 = 1.40

```

```

aX1 = 160
aX2 = 70

```

```
aH1 = 100  
aH2 = 140  
aH3 = 65
```

```
dH = 60  
END
```

```
ATOMS   Z-Matrix
```

```
1. C    0 0 0 0.    0.    0.  
2. Hg   1 0 0 rHg   0.    0.  
3. Cl   2 1 0 rX1   aX1   0.  
4. H    1 2 3 rH1   aH1   dH  
5. H    1 2 3 rH1   aH1  -dH  
6. H    1 2 3 rH2   aH2   180.  
7. H    1 2 3 rH3   aH3   180.  
8. Cl   2 1 3 rX2   aX2   180.
```

```
END
```

```
FRAGMENTS
```

```
Hg  t21.Hg  
C   t21.C  
H   t21.H  
Cl  t21.Cl
```

```
END
```

```
endinput  
eor
```

For the density-functional the Local Density approximation is used (no GGA corrections), with a correlation correction term due to Stoll (see the User's Guide).

At each geometry cycle the interatomic distance matrix is printed (print atdist).

The initial geometry is a reasonable but not very accurate estimate of the Transition State. The program needs quite a few cycles to converge, which is rather typical for ts searches: they are a lot more tricky and fail more often than a simple minimization.

## H<sub>2</sub>O: constraint Linear Transit

*Sample directory:* adf/e\_LT\_constraint/

The CONSTRAINT keyword allows geometry optimizations with constraints defined by arbitrary linear combinations of (internal) coordinates. The constraint has to be satisfied at the start of the geometry optimization.

Example for bond length constraint, where at the start of the linear transit  $r_{OH1}=R1=1.0$ , and  $r_{OH2}=R2=1.5$ , such that  $(-1.0)*R1+(1.0)*R2=0.5$ , and in the final geometry  $-R1+R2=0.0$  (Reactcoord 0.5 0.0)

```
$ADFBIN/adf << eor
title constraint keyword
XC
  GGA Becke Perdew
END
```

```
Geometry
LinearTransit 6
End
```

```
Integration 5.0
```

```
Atoms Internal
O 0 0 0
H 1 0 0 R1
H 1 2 0 R2 109.9
End
```

```
GeoVar
R1 =1.
R2 =1.5
End
```

```
Constraint
ReactCoord 0.5 0.0
R1 -1.0
R2 1.0
SubEnd
End
```

```
Basis
Type DZP
End
```

```
end input
eor
```

## Analysis options: fragment orbitals and bond energy decomposition

### Ni(CO)<sub>4</sub>: Compound Fragments

*Sample directory:* adf/e\_Frags\_NiCO4/

An illustration of the fragment feature of adf

A transition metal complex is built from a Nickel atom and four CO fragments. The outcomes allows for an analysis (of molecular orbitals and the Bonding energy) in terms of the fragment properties. It is a Single Point calculation. Geometry optimization would not have been possible in this set-up because an optimization requires that only single-atom fragments are used.

The three atoms are created first: C, O, and Ni. For Carbon and Oxygen a type-DZ basis set is used (double-zeta) using the Basis key, while Ni gets a type-TZP basis (triple-zeta plus polarization).

### CO

The CO molecule, to serve as a fragment template in Ni(CO)<sub>4</sub>, is computed from the atomic fragments C and O. The coordinate values (atoms) are in bohr, rather than in Angstrom because the unit-of-length is redefined by the key units with subkey length.

The key scf is used to specify a somewhat tighter convergence criterion than the default, just to illustrate how to do this (normal settings are quite adequate).

The TAPE21 result file is renamed *t21.CO*.

```
$ADFBIN/adf <<eor
title CO (as fragment for NiCO4)
```

```
SCF
converge 1e-8
end
```

```
units
length bohr
end
```

```
atoms
C 0 0 0
O 0 0 2.15617844
end
```

```
Basis
  Type DZ
  Core Small
End
```

```
endinput
```

```
eor
```

```
mv TAPE21 t21.CO
```

## Main calculation

Apart from the title, the input contains comment. This does not specify computational parameters but is only echoed in the output header, similar to the title. Contrary to the title, however, such comments are not preserved, apart from their echo in output and they are not written to TAPE21 or any other result file.

The atomic coordinates (atoms) are given in bohr (Units). To supply the numerical values use is made of user-defined constants (define): xyzC and xyzOx. This is convenient and it prevents typing errors when several coordinate values are identical, in particular when they carry a lot of decimal places.

The Atoms records contain also a specification of the fragments to which the respective atoms belong: four different CO fragments. No fragment is specified for the Ni atom, which implies that it is a fragment on its own.

The numbers at the very left of the records (1 through 9, with (optionally) a period after them), have no relevance. You can set them for ease of reference or counting.

```
$ADFBIN/adf <<eor
```

```
title Ni(CO)4, from fragments Ni and CO
```

```
COMMENT
```

```
No geometry optimization possible, because not all fragments  
are single atoms
```

```
END
```

```
units
```

```
length bohr
```

```
end
```

```
DEFINE
```

```
xyzC=2.0053211
```

```
xyzOx=3.2501913
```

```
END
```

```
atoms
```

```
1. Ni 0 0 0
```

```
2. C xyzC xyzC xyzC f=CO/1
```

```
3. C -xyzC -xyzC xyzC f=CO/2
```

```
4. C xyzC -xyzC -xyzC f=CO/3
```

```
5. C -xyzC xyzC -xyzC f=CO/4
```

```
6. O xyzOx xyzOx xyzOx f=CO/1
```

```
7. O -xyzOx -xyzOx xyzOx f=CO/2
```

```
8. O xyzOx -xyzOx -xyzOx f=CO/3
```

```
9. O -xyzOx xyzOx -xyzOx f=CO/4
```

```
end
```

```
fragments
```

```
CO t21.CO
```

```
Ni t21.Ni
```

```
end
```

endinput  
eor

## PtCl<sub>4</sub>H<sub>2</sub><sup>2-</sup>: Fragments again

Sample directory: `adf/e_Frags_PtCl4H2`

The (scalar) relativistic option (Pauli formalism) is used because of the presence of the relativistic Pt atom. The complex is built from fragments H<sub>2</sub> and PtCl<sub>4</sub><sup>2-</sup>.

### Dirac: relativistic core potentials

The program *dirac* is applied to generate the corepotentials file for all involved atom types, including Hydrogen. The latter has no frozen core, let alone a relativistic one, but the corepotentials file also contains the total (relativistic) atomic potential. The (relativistic) atomic total potential is used in some types of relativistic options only, but it is a good idea to simply always run DIRAC for all the atoms whenever you do a relativistic calculation.

```
$ADFBIN/dirac <$adfresources/Dirac/Cl.2p
$ADFBIN/dirac <$adfresources/Dirac/Pt.4d
$ADFBIN/dirac <$adfresources/Dirac/H
```

```
mv TAPE12 t12.rel
```

The script above generates *one* TAPE12 file. The second and third dirac runs recognize the presence of the TAPE12 file (with the standard name 'TAPE12') that resulted from the earlier ones and they write their resulting data to the tail of it.

### Basic atoms, non-default settings

```
$ADFBIN/adf <<eor
Create H file=$ADFRESOURCES/DZP/H
XC
  LDA vwn
  GGA becke perdw
End
```

```
Relativistic Scalar
CorePotentials t12.rel &
  H 3
End
End Input
eor
```

```
mv TAPE21 t21H
```

The final calculations of the molecule and larger fragments are performed with gga ('NonLocal') xc corrections. Although it is not necessary to use the same settings in the Create runs, applying them looks 'nicer' and gives a better approximation of the bond energy of the molecule with respect to the basic atoms. Here it serves to show that also in a Create run various options can be used.

```
$ADFBIN/adf <<eor
```

```

create Cl file=$ADFRESOURCES/DZP/Cl.2p
xc
  lda vwn
  GGA becke perdew
end

relativistic scalar
corepotentials t12.rel &
  Cl 1
end

end input
eor

mv TAPE21 t21Cl

```

```

$ADFBIN/adf <<eor
Create Pt file=$ADFRESOURCES/DZ/Pt.4d
XC
  lda vwn
  GGA becke perdew
End

```

```

Relativistic scalar
CorePotentials t12.rel &
  Pt 2
End

```

```

End Input
eor

```

```

mv TAPE21 t21Pt

```

It is important to use the relativistic option in the creation of the fragments if the final molecule will use it as well. The corepotentials file is attached and the input indicates that the section on that file for Cl is #1, while the Pt data are in section #2.

## Fragments H<sub>2</sub> and PtCl<sub>4</sub><sup>2-</sup>

Now, all basic atoms have been generated. We go on to generate the two larger fragments H<sub>2</sub> and PtCl<sub>4</sub><sup>2-</sup> from which we are going to build the final complex.

```

$ADFBIN/adf <<eor
Title H2 R=1.68a.u.
NoPrint sfo,frag,functions

```

```

Units
  length bohr
End

```

```

Atoms
H    0.0      0.0      0.84

```

```
H 0.0 0.0 -0.84
```

```
End
```

```
Fragments
```

```
H t21H
```

```
End
```

```
XC
```

```
lda vwn
```

```
GGA becke perdew
```

```
End
```

```
Relativistic Scalar
```

```
CorePotentials t12.rel &
```

```
H 3
```

```
End
```

```
End Input
```

```
eor
```

```
mv TAPE21 t21H2
```

The result file TAPE21 is renamed and saved to serve as fragment file.

```
$adf <<eor
```

```
title PtCl4 (2-)
```

```
noprint sfo,frag,functions
```

```
units
```

```
length bohr
```

```
end
```

```
ATOMS
```

```
Pt 0 0 0
```

```
Cl 4.361580 0.000000 0
```

```
Cl 0.000000 4.361580 0
```

```
Cl -4.361580 0.000000 0
```

```
Cl 0.000000 -4.361580 0
```

```
end
```

```
fragments
```

```
Pt t21Pt
```

```
Cl t21Cl
```

```
end
```

```
xc
```

```
lda vwn
```

```
GGA becke perdew
```

```
end
```

```
relativistic scalar
```

```
corepotentials t12.rel &
```

```
Cl 1
```

```
Pt 2
```

```
end
```

```
charge -2
```

```
end input
```

```
eor
```

```
mv TAPE21 t21PtCl4
```

The key charge is used to specify the net total charge. The default for the net total charge is the sum-of-fragment-charges. The fragments (Pt and Cl atoms) have been computed neutrally, but we want to calculate the PtCl<sub>4</sub> complex as a 2- ion.

## Main calculation

Finally we compute PtCl<sub>4</sub>H<sub>2</sub><sup>2-</sup> from the fragments PtCl<sub>4</sub><sup>2-</sup> and H<sub>2</sub>.

```
$ADFBIN/adf <<eor
```

```
title PtCl4 H2
```

```
units
```

```
length bohr
```

```
end
```

```
integration 4.0
```

```
xc
```

```
lda vwn
```

```
  GGA becke perdew
```

```
end
```

```
relativistic scalar
```

```
corepotentials t12.rel &
```

```
H 3
```

```
Cl 1
```

```
Pt 2
```

```
end
```

```
ATOMS
```

```
Pt 0      0      0      f=PtCl4
```

```
Cl 4.361580 0.000000 0.00000000 f=PtCl4
```

```
Cl 0.000000 4.361580 0.00000000 f=PtCl4
```

```
Cl -4.361580 0.000000 0.00000000 f=PtCl4
```

```
Cl 0.000000 -4.361580 0.00000000 f=PtCl4
```

```
H 0.0      0.0      5.58      f=H2
```

```
H 0.0      0.0      7.26      f=H2
```

```
end
```

```
fragments
```

```
PtCl4  t21PtCl4
```

```
H2  t21H2
```

```
end
```

```
end input  
eor
```

Note that, although the key charge is not supplied, the molecule is *not* neutral: the default charge (that is, omitting the keys charge, occupations) is the *sum-of-fragments*: the fragments here are H<sub>2</sub> and PtCl<sub>4</sub><sup>2-</sup>, yielding a net charge for the molecule of minus two.

Note the f= fragment specification in the Atoms block. No fragment-numbering suffix (/n) is required because there is only one fragment of each fragment *type*.

## H<sub>2</sub>: Spin-unrestricted Fragments

Sample directory: `adf/e_UnrFrag_H2`

This is a small but important example to illustrate what goes into an accurate calculation of the 'true' bond energy of a molecule. The (ADF-specific) problem is that in a straightforward molecular calculation, the bond energy is computed as the energy difference between at the one hand the molecule, and at the other hand the isolated *spherically symmetric spin-restricted* atoms. The *italic*-typed features imply that the reference (comparison) state is usually not the physical ground state of the reference system (isolated atoms) and hence the computed energy difference has no direct relation to experimental data. To account for the true atomic ground states, one has to add correction terms. Study this sample carefully to make sure that you fully understand the steps to take and consult the User's Guide for details. See also the Theory document for a discussion of multiplet states.

See also the example, `SD_Cr(NH3)6`.

The H<sub>2</sub> case consists of a sequence of simple calculations to demonstrate the Unrestricted Fragments option. The energy difference between an unrestricted fragment as it is used in `adf` and a *self-consistent* unrestricted fragment is also computed. This turns out to be quite small, confirming that the `adf` approach, although not formally exact, is adequate for practical purposes.

```
$ADFBIN/adf <<eor
create H file=$ADFRESOURCES/DZP/H
end input
eor

mv TAPE21 t21H

$ADFBIN/adf <<eor
title H unrestr., not self-consistent (as used in unr.frag. calcs)

scf
iterations 0 ! prohibit relaxation
end

unrestricted
charge 0 1 ! if not specified up and down electrons
! will both get 0.5 electron: in fact restricted

fragments
H t21H
end

atoms
H 0 0 0
end

endinput
eor

rm TAPE21 logfile
```

By setting the scf iterations to zero (a value of one (1) would give the same result) we prevent cycling to self-consistency. The energy of the 'final' one-electron orbitals is consequently computed in the start-up potential, i.e. the field of the restricted (basic) atom, where spin- $\alpha$  and spin- $\beta$  are equally occupied, in this case by 0.5 electron each. The not-self-consistent, unrestricted H atom is precisely the 'unrestricted' fragment as it can be used in an adf calculation with unrestricted fragments. The fragment file must be the TAPE21 result file from a *restricted* run, but at start-up you can specify that the Fragment Orbitals are, for purposes of reference and comparison, occupied in an unrestricted way in the final molecule.

A calculation that uses *restricted* fragments right away computes the bonding energy relative to the restricted fragments. The difference between using restricted and unrestricted fragments is the 'bonding' energy computed in the run above.

```
$ADFBIN/adf <<eor
title H unr. self-consistent from unr.0
```

```
unrestricted
charge 0 1
```

```
fragoccupations
H
s 1 // 0
subend
end
```

```
Atoms
H 0 0 0
end
```

```
fragments
H t21H
end
```

```
end input
eor
```

```
rm TAPE21 logfile
```

Here we start with the unrestricted fragment and relax to self-consistency. The 'bonding energy', i.e. the relaxation energy, is very small, demonstrating that using non-self-consistent unrestricted fragments involves only a small error (which, moreover, can be computed as shown here).

The key UnRestricted sets the spin-unrestricted mode. The key Charge is used to specify a net total charge of zero and a net total spin polarization by an excess of 1.0 spin- $\alpha$  electrons over spin- $\beta$ .

```
$ADFBIN/adf <<eor
title H2 restricted, from restricted fragments
```

```
ATOMS
H 0 0 0.375
H 0 0 -0.375
end
```

```
fragments
H t21H
end
```

```
end input
eor
```

```
rm TAPE21 logfile
```

This is the simplest approach, using *restricted* fragments. The bonding energy must be corrected because the reference (restricted H atoms, with 0.5 electrons in spin- $\alpha$  and 0.5 in spin- $\beta$ ) is far from the true H-atom ground state: see the previous runs on the single H atom.

```
$ADFBIN/adf <<eor
title H2 from unrestricted fragments
```

```
ATOMS
H.1 0 0 0.375
H.2 0 0 -0.375
end
```

```
fragments ! two different fragment types are necessary
!         because the two atoms get different FragOccupations
!         (see below), while the key FragOc.. addresses
!         only fragmentTYPES
```

```
H.1 t21H
H.2 t21H
end
```

```
charge 0
```

```
occupations
  sigma 2 ! specify the state (not always
          ! necessary)
end
```

```
fragoccupations
H.1
  s 1 // 0
subend
H.2
  s 0 // 1
subend
end
```

```
modifystartpotential
H.1 1 // 0 ! this helps SCF start-up
H.2 0 // 1 ! but is here not necessary
end
```

```
end input
eor
```

```
rm TAPE21 logfile
```

This should be a fair approximation (in the *lda* model) to the bonding energy of H<sub>2</sub> with respect to the unrestricted H atoms. The difference between the bonding energies of this and the previous run should be very close to the energy of the not-self-consistent unrestricted H-atom with respect to the restricted basic atom (calculation #2).

## Excited state

```
$ADFBIN/adf <<eor  
title H2 excited
```

```
ATOMS
```

```
H 0.0 0.375  
H 0.0 -0.375  
end
```

```
fragments
```

```
H t21H  
end
```

```
fragoccupations
```

```
H  
s 1 // 0  
subend  
end
```

```
unrestricted
```

```
charge 0 2
```

```
occupations
```

```
sigma.g 1 // 0  
sigma.u 1 // 0  
end
```

```
end input  
eor
```

Finally the calculation of an excited state, with respect to unrestricted fragments. The excitation energy is obtained by comparing the energy with the energy of the ground state calculation. This difference compares reasonably, but not accurately, to the difference in one-electron ground state energies of the involved orbitals (Koopman's theorem).

Note that excitation energies can also be calculated with Time-Dependent DFT, using the RESPONSE module of ADF. See related sample runs.

## PCCP: Bond Energy analysis open-shell fragments

*Sample directory:* adf/e\_PCCP\_Unr\_BondEnergy/

This example illustrates advanced usage of the bond energy decomposition scheme used in ADF.

A proper decomposition of an electron-pair bond energy requires specifying opposite spins for the unpaired electrons of the respective radical fragments, which can be done with the input key FragOccupations. The specified alpha- and beta-spin configurations of the radical fragments are shown in the output section B U I L D.

Please note that if one neglects explicitly specifying opposite spins for the unpaired electrons of the fragments, each of them is treated as being half an alpha and half a beta electron and consequently, they enter into a spurious Pauli repulsive interaction. This results, amongst others, into the Pauli repulsion term being too repulsive and the orbital interaction term being too much stabilizing.

The example consists of an analysis of the C-C single bond between two CP radicals in the four-atomic molecule PCCP. The CP fragment calculations used to provide the TAPE21 for the overall PCCP calculation must be done, for technical reasons, in the restricted mode ("cp\_fpccp\_asr"). The proper spins are then specified in the calculation of the overall molecule using the FragOccupations key ("pccp\_fa1\_as"). Note that this implies a slight approximation because the bond energy computed in this way refers to the energy difference between closed-shell PCCP and two CP radicals that are described by orbitals from a spin-restricted SCF calculation, which have been given an unrestricted occupation. In other words, the set of alpha- and beta-spin orbitals are identical and the effect of spin polarization is missing. In practice, this leads to minor energy differences with respect to the correct bond energy, that is, the energy difference between closed-shell PCCP and two CP radicals treated in the unrestricted mode, i.e., for which the set of alpha- and beta-spin orbitals are allowed to relax toward different solutions in the SCF procedure. This correction term can be computed directly by carrying out

a an unrestricted computation of the CP radical ("cp\_fpccp\_asu") using the restricted CP radical ("cp\_fpccp\_asr") as a fragment.

```
$ADFBIN/adf<<eor
TITLE cp_fpccp_asr
```

```
XC
GRADIENTS BECKE PERDEW
END
```

```
ATOMS
  C   .0000  .0000  .6681
  P   .0000  .0000  2.2555
END
```

```
FRAGMENTS
C t21.C
P t21.P
END
```

```
integration 5.0
```

```
END INPUT
eor
```

```
mv TAPE21 t21cp_fpccp
```

```
$ADFBIN/adf<<eor  
TITLE cp_fpccp_asu
```

```
XC  
GRADIENTS BECKE PERDEW  
END
```

```
ATOMS  
C .0000 .0000 .6681 f=CP  
P .0000 .0000 2.2555 f=CP  
END
```

```
FRAGMENTS  
CP t21cp_fpccp  
END
```

```
UNRESTRICTED  
CHARGE 0 1
```

```
integration 5.0
```

```
END INPUT  
eor
```

```
rm TAPE21 logfile
```

```
$ADFBIN/adf<<eor  
TITLE pccp_fa1_as
```

```
EPRINT  
ORBPOP 20 20  
SUBEND  
END
```

```
XC  
GRADIENTS BECKE PERDEW  
END
```

```
ATOMS  
P .0000 .0000 2.2555 f=CP_A  
C .0000 .0000 .6681 f=CP_A  
C .0000 .0000 -.6681 f=CP_B  
P .0000 .0000 -2.2555 f=CP_B  
END
```

```
integration 5.0
```

```
FRAGMENTS  
CP_A t21cp_fpccp  
CP_B t21cp_fpccp  
END
```

SYMMETRY C(LIN)

FRAGOCCUPATIONS

CP\_A

SIGMA 3//2

PI 2//2

SUBEND

CP\_B

SIGMA 2//3

PI 2//2

SUBEND

END

END INPUT

eor

## TIH: Spin-Orbit SFO analysis

*Sample directory:* `adf/e_TIH_SO_analysis/`

Application of the Spin-Orbit relativistic option (using double-group symmetry) to TIH with a detailed analysis of the spinors in terms of SFOs (Symmetrized Fragment Orbitals).

In order to get the population analysis, one should have one scalar relativistic fragment, which is the whole molecule. The SFOs in this case are the scalar relativistic orbitals, which are already orthonormal, because one has only one fragment which is the whole molecule.

First the relativistic fragment is made, including the create of the atoms:

```
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/TI
$ADFBIN/dirac -n1 < $ADFRESOURCES/Dirac/H
mv TAPE12 t12.rel
```

```
$ADFBIN/adf <<eor
create TI file=$ADFRESOURCES/ZORA/TZ2P/TI
xc
  LDA vwn
  GGA becke perdew
end
relativistic scalar zora
corepotentials t12.rel &
TI 1
H 2
end
end input
eor
mv TAPE21 t21.TI
```

```
$ADFBIN/adf <<eor
create H file=$ADFRESOURCES/ZORA/TZ2P/H
xc
  LDA vwn
  GGA becke perdew
end
relativistic scalar zora
corepotentials t12.rel &
TI 1
H 2
end
end input
eor
mv TAPE21 t21.H
```

```
$ADFBIN/adf <<eor
title TIH, scalar relativistic zora
```

```
integration 6.0
```

```

relativistic scalar zora
corepotentials t12.rel &
TI 1
H 2
end

```

```

ATOMS
TI 0.0 0.0 0.0
H 0.0 0.0 1.870
end

```

```

fragments
TI t21.TI
H t21.H
end

```

```

xc
LDA vwn
GGA becke perdew
end

```

```

end input
eor

```

```

mv TAPE21 t21.TIH

```

In order to get the population analysis, one should have one scalar relativistic fragment, which is the whole molecule, which is TIH in this case.

```

$ADFBIN/adf <<eor
title TIH from fragment TIH, with SpinOrbit coupling

```

```

integration 6.0

```

```

relativistic spinorbit zora
corepotentials t12.rel &
TI 1
H 2
end

```

```

ATOMS
TI 0.0 0.0 0.0 f=TIH
H 0.0 0.0 1.870 f=TIH
end

```

```

fragments
TIH t21.TIH
end

```

```

xc
LDA vwn
GGA becke perdew
end

```

```
end input
eor
```

The output gives something like:

```
=====
Double group symmetry : *** J1/2 ***
=====

=== J1/2:1 ===
```

Spinors expanded in SFOs

```
....
Spinor:      21      22      23      24
occup:      1.00    1.00    1.00    0.00
-----
SFO SIGMA
13.alpha:  0.7614+0.0000i  0.0096+0.0000i  0.0052+0.0000i  -0.0006+0.0000i
14.alpha:  0.0154+0.0000i -0.9996+0.0000i  0.0208+0.0000i  -0.0077+0.0000i
15.alpha: -0.0146+0.0000i  0.0185+0.0000i  0.9849+0.0000i  0.1625+0.0000i
SFO PI:x
8.beta :   0.4578+0.0000i  0.0091+0.0000i  0.0112+0.0000i  0.0030+0.0000i
9.beta :   0.0005+0.0000i -0.0074+0.0000i -0.1119+0.0000i  0.6910+0.0000i
SFO PI:y
8.beta :   0.0000+0.4578i  0.0000+0.0091i  0.0000+0.0112i  0.0000+0.0030i
9.beta :   0.0000+0.0005i  0.0000-0.0074i  0.0000-0.1119i  0.0000+0.6910i
....
```

Left out are a lot of small numbers. The meaning is that a spinor of  $J_z=1/2$  symmetry can have SIGMA and PI character, for example, the 21st spinor with occupation number 1.0, is approximately  
(21  $J_z=1/2$ ) = 0.76 (13 SIGMA alpha) + 0.46 (8 PI:x beta) + i 0.46 (8 PI:y beta)

Next in the SFO contributions per spinor the real and imaginary spin alpha part and real and imaginary spin beta part are all summed together to give a percentage of a certain SFO. are summed. For example the 21st spinor has almost 60% (13 SIGMA) character.

```
SFO contributions (%) per spinor
Spinor:   21  22  23  24
occup:   1.00 1.00 1.00 0.00
-----
SFO SIGMA
13:  57.97  0.01  0.00  0.00
14:   0.02 99.92  0.04  0.01
15:   0.02  0.03 97.01  2.64
SFO PI:x
8:  20.96  0.01  0.01  0.00
9:   0.00  0.01  1.25 47.75
SFO PI:y
8:  20.96  0.01  0.01  0.00
9:   0.00  0.01  1.25 47.75
```

## Various applications with ADF

### Cr(NH<sub>3</sub>)<sub>6</sub>: Multiplet States

*Sample directory:* adf/e\_SD\_CrNH3\_6/

The computation of multiplet states corresponding to an open-shell system can be carried out with ADF by first computing the 'Average-of-Configuration' (aoc) state, where all orbitals in the open shell are degenerate and equally occupied. This computation is spin-restricted and serves as a fragment file for the multiplet run, where then different occupation numbers are assigned to the various orbitals in the open shell. The corresponding energies are computed in the field of the aoc, which is achieved by *not* iterating the self-consistency equations to convergence but only computing the orbitals in the initial field.

Since ADF requires that all symmetry-partners in an irreducible representation (irrep) have equal occupations, the multiplet calculation, where such orbitals are *not* equally occupied, must be carried out in a formally lower pointgroup symmetry. The pointgroup to select and the appropriate occupation numbers to apply must be worked out by the user 'on paper' in advance. An auxiliary program asf, developed by the group of Claude Daul in Fribourg can be used to determine which calculations are needed, and how to compute the multiplet energies from the results. See the discussion of Multiplet energies in the Theory document.

The script starts with the 'creation' of the required basic atoms, N, H, Cr using a fair basis set quality.

The next step is the computation of the ammonia fragment NH<sub>3</sub>. This is not a crucial step here: the multiplet state computation can equally well be carried out by not using any intermediate compound fragments. However, it illustrates once more how a bigger molecule can be built up from smaller, but not trivial fragments.

```
$ADFBIN/adf <<eor
title AMMONIA
NOPRINT sfo,frag,functions
```

```
define
xH=0.95522523
yH=xH*sqrt(3)/2
zH=0.3711068
end
```

```
atoms
N 0 0 0
H -xH 0 zH
H xH/2 -yH zH
H xH/2 yH zH
end
```

```
Basis
Type TZP
Core Small
End
```

```
symmetry C(3V)
```

```
endinput
eor
```

```
mv TAPE21 t21.NH3
```

The input of the atomic coordinates uses expressions, in this case to enforce exact symmetry relations that would otherwise require 14-digit input values or some inaccuracy. The symmetry specification is redundant: the program would also find it by itself.

## Average-of-Configuration

The next step is to compute the reference state, with respect to which we will later compute the multiplet states. The reference state is the so-called 'Average-of-configuration' (aoc) state. The result file (TAPE21) of this calculation will be used as a fragment file.

```
$ADFBIN/adf <<eor
title Cr(NH3)6 : Average-of-Configuration run
```

```
COMMENT
using NH3-fragments
END
```

```
symmetry D(3d)
```

```
scf
iterations 25
mix 0.15
end
```

```
atoms
Cr  0.000000  0.000000  0.000000
N   0.000000  1.714643  1.212436 f=NH3/1
H   0.000000  1.466154  2.206635 f=NH3/1
H  -0.827250  2.293404  1.036727 f=NH3/1
H   0.827250  2.293404  1.036727 f=NH3/1
N  -1.484924 -0.857321  1.212436 f=NH3/2
H  -1.269726 -0.733077  2.206635 f=NH3/2
H  -1.572521 -1.863121  1.036727 f=NH3/2
H  -2.399771 -0.430282  1.036727 f=NH3/2
N   1.484924 -0.857321  1.212436 f=NH3/3
H   1.269726 -0.733077  2.206635 f=NH3/3
H   2.399771 -0.430282  1.036727 f=NH3/3
H   1.572521 -1.863121  1.036727 f=NH3/3
N   0.000000 -1.714643 -1.212436 f=NH3/4
H   0.000000 -1.466154 -2.206635 f=NH3/4
H   0.827250 -2.293404 -1.036727 f=NH3/4
H  -0.827250 -2.293404 -1.036727 f=NH3/4
N   1.484924  0.857321 -1.212436 f=NH3/5
H   1.269726  0.733077 -2.206635 f=NH3/5
H   1.572521  1.863121 -1.036727 f=NH3/5
H   2.399771  0.430282 -1.036727 f=NH3/5
N  -1.484924  0.857321 -1.212436 f=NH3/6
H  -1.269726  0.733077 -2.206635 f=NH3/6
```

```

H -2.399771 0.430282 -1.036727 f=NH3/6
H -1.572521 1.863121 -1.036727 f=NH3/6
H -1.572521 1.863121 -1.036727 f=NH3/6
end

```

```

fragments
Cr t21.Cr
NH3 t21.NH3
end

```

```

occupations
A1.G 8.75
A2.G 2
E1.G 16 1.5 0.75
A1.U 2
A2.U 8
E1.U 20
END

```

```

end input
eor

```

```
mv TAPE21 t21.CrA6ES
```

Occupation numbers are specified, to make certain what the reference state is that we will start from in the subsequent calculations. The result file TAPE21 is saved to serve as fragment file in the subsequent calculations.

## One-determinant states

Now, we proceed with the multiplet calculations. In the example they are combined in one single run, but they could also be evaluated in separate runs. For each calculation it is required to:

- Use the aoc TAPE21 file as fragment file
- Choose which molecular orbitals in the open shell to occupy: select the appropriate pointgroup symmetry and the UnRestricted key if necessary and specify the occupation numbers, using the irreducible representations of the selected point group.

The results are one-determinant calculations, which must then, later, be combined analytically to obtain the required multiplet energy values.

```

$ADFBIN/adf <<eor
title Cr(NH3)6 : SlaterDeterminants run
NOPRINT frag

```

```
symmetry C(I) ! lower symmetry
```

```

scf
iterations 0
end

```

```

atoms
Cr 0.000000 0.000000 0.000000 f=CrA6
N 0.000000 1.714643 1.212436 f=CrA6

```

```

H    0.000000    1.466154    2.206635 f=CrA6
H   -0.827250    2.293404    1.036727 f=CrA6
H    0.827250    2.293404    1.036727 f=CrA6
N   -1.484924   -0.857321    1.212436 f=CrA6
H   -1.269726   -0.733077    2.206635 f=CrA6
H   -1.572521   -1.863121    1.036727 f=CrA6
H   -2.399771   -0.430282    1.036727 f=CrA6
N    1.484924   -0.857321    1.212436 f=CrA6
H    1.269726   -0.733077    2.206635 f=CrA6
H    2.399771   -0.430282    1.036727 f=CrA6
H    1.572521   -1.863121    1.036727 f=CrA6
N    0.000000   -1.714643   -1.212436 f=CrA6
H    0.000000   -1.466154   -2.206635 f=CrA6
H    0.827250   -2.293404   -1.036727 f=CrA6
H   -0.827250   -2.293404   -1.036727 f=CrA6
N    1.484924    0.857321   -1.212436 f=CrA6
H    1.269726    0.733077   -2.206635 f=CrA6
H    1.572521    1.863121   -1.036727 f=CrA6
H    2.399771    0.430282   -1.036727 f=CrA6
N   -1.484924    0.857321   -1.212436 f=CrA6
H   -1.269726    0.733077   -2.206635 f=CrA6
H   -2.399771    0.430282   -1.036727 f=CrA6
H   -1.572521    1.863121   -1.036727 f=CrA6

```

end

fragments

CrA6 t21.CrA6ES

end

UnRestricted

SlaterDeterminants

Check AOC

```

A1.g  4 0.375      // 4 0.375
A2.g  1           // 1
E1.g:1 4 0.375 0.1875 // 4 0.375 0.1875
E1.g:2 4 0.375 0.1875 // 4 0.375 0.1875
A1.u  1//1
A2.u  4//4
E1.u:1 5//5
E1.u:2 5//5

```

SUBEND

State1

```

A1.g  4 1          // 4 1
A2.g  1           // 1
E1.g:1 4 0 0      // 4 0 1
E1.g:2 4 0 0      // 4 0 0
A1.u  1//1
A2.u  4//4
E1.u:1 5//5
E1.u:2 5//5

```

SUBEND

State2

```

A1.g  4 1          // 4 1
A2.g  1           // 1

```

```

E1.g:1 4 0 0 // 4 1 0
E1.g:2 4 0 0 // 4 0 0
A1.u 1//1
A2.u 4//4
E1.u:1 5//5
E1.u:2 5//5
SUBEND
State3
A1.g 4 1 // 4 1
A2.g 1 // 1
E1.g:1 4 0 1 // 4 0 0
E1.g:2 4 0 0 // 4 0 0
A1.u 1//1
A2.u 4//4
E1.u:1 5//5
E1.u:2 5//5
SUBEND
end

end input
eor

```

The SlaterDeterminants block may contain any number of sub blocks, each starting with an (arbitrary) title record, followed by a set of occupation numbers and closed by a SubEnd record. Each such subkey block specifies a single one-determinant-state calculation. All occupation numbers must reference the irreps of the specified pointgroup symmetry, C(I) in the example, and must be just a reassignment of the electrons that are equally distributed over the corresponding degenerate irreps in the reference aoc calculation.

The so-obtained energies of the one-determinant states can now be combined to calculate the desired multiplet energies. See the Theory document and the adf User's Guide.

Note carefully that in the calculation of the SingleDeterminants, the scf procedure is prevented to cycle to convergence by setting the subkey Iterations to zero in the SCF data block.

## Cr(CO)<sub>5</sub>+CO: Basis Set Superposition Error

Sample directory: `adf/e_BSSE_CrCO6/`

A study of the Basis Set Superposition Error (BSSE) in the formation of Cr(CO)<sub>6</sub> from CO and Cr(CO)<sub>5</sub>.

The basis set superposition error (BSSE) can be calculated with the help of the option to create *Alternative Chemical Elements*. or *Ghost* atoms.

An alternative chemical element is an element with a special feature, not corresponding to one of the predefined chemical elements. It may have, for instance, a different effective nuclear charge or a 'special' atomic mass.

For the BSSE calculation special chemical elements must be created to describe the 'ghost' atoms, which have zero nuclear charge and mass. They do have basis functions (and fit functions), however, and they are used to calculate the lowering of the energy of the system to which the ghost atoms are added, due to the enlargement of the basis by the ghost basis. The ghost atom has the same basis and fit set as the normal element but no nuclear charge and no frozen core (there must be no core description in the Create data file for the ghost atom!).

The following calculations are carried out:

- 1. CO from C and O. This yields the bond energy of CO with respect to the (restricted) basic atoms.
- 2. CO from the fragments CO (as calculated in 1) and the ghost atom Cr and 5 Carbon and 5 Oxygen ghost atoms. The ghost atomic fragments provide basis and fit functions but do not contribute charge or potential to the molecule. The bond energy of this calculation is the energy lowering of CO due to the additional basis functions. This is the BSSE for CO.
- 3. Cr(CO)<sub>5</sub> from Cr and 5 CO's. This yields the ('normal') bond energy with respect to the given fragments.
- 4. Cr(CO)<sub>5</sub> from Cr(CO)<sub>5</sub> as fragment (as calculated in 3) but with the CO basis functions added on the position of the 6th CO ('ghost' CO). The bond energy is the BSSE for Cr(CO)<sub>5</sub>.
- 5. Cr(CO)<sub>6</sub> with Cr(CO)<sub>5</sub> and CO as fragments. The bond energy is the one without BSSE. This bond energy can now be corrected by the sum of superposition contributions of calculations 2 and 4.

This series of calculations is carried out with basis set DZ.

Next, the two BSSE runs (#2 and #4 in the list above) are repeated, but now with the core orthogonalization functions omitted from the ghost bases. One may argue about whether these functions should be included in the ghost basis sets, but since they are very contracted around the ghost nuclei they are not expected to contribute significantly anyway and may then just as well be omitted. This is explicitly verified in the current example by demonstrating that the BSSE is not significantly affected by omitting these functions.

Finally, the whole thing might be redone with basis set TZP, to show that the BSSE decreases with larger basis.

The calculations for the type DZ basis are contained in the sample script (with input- and output files). Those for type TZP bases can be set up easily and may be done as an exercise.

For the first series of calculations, with basis type DZ, the input files are discussed below and the relevant parts are echoed from the output files where the energy decomposition and the total bond energy are

printed.

For the other series, using type TZP basis sets, only a summary of the results is given.

## Computational details

The calculations in this example all use:

- 1. Frozen core level for the Chromium atom: 2p (for Carbon and Oxygen: 1s);
- 2. Numerical integration precision 4.0 (in Create runs 10.0, the default);
- 3. Default settings for model parameters such as density functional (key XC) and for the remaining computational settings

## Basis DZ, including Core Functions

### Creation of ghost atoms

Ghost atoms must be created like normal chemical elements. The adf database does not provide the ghost database files. They are easily constructed from the normal database file of the pertaining chemical element: only the frozen core references have to be adapted such that the ghost atom will not have a frozen core. This affects the sections 'core' and 'description' in the database file (see the User's Guide).

For the creation of the Carbon ghost atom with basis DZ the database file is:

Carbon (II, ghost)

BASIS

1S 5.40

2S 1.24

2S 1.98

2P 0.96

2P 2.20

END

CORE 0 0 0 0

END

DESCRIPTION

END

FIT

1S 10.80

2S 11.59

2S 7.59

2S 4.97

3S 4.79

3S 3.35

3S 2.34

3S 1.64

2P 8.34

2P 5.14

```

3P 4.67
3P 3.10
3P 2.06
3D 5.88
3D 3.84
3D 2.51
3D 1.64
4F 5.40
4F 3.55
5G 4.50
END

```

```

FITCOEFFICIENTS
/
END

```

Observe that there are four integers zero after the keyword core, indicating that there are no s-, p-, d-, or f-type frozen core shells. Specification of any frozen core shells would imply the insertion of (core) electrons around the ghost atoms in the calculation.

Consequently, the data block directly below core is empty: no Slater-type functions are required to describe any frozen core orbitals.

Finally, the description data block is empty: no expansion coefficients that would describe the frozen core orbitals in terms of the Slater-type expansion functions.

All other data (apart from the title, which is just a label) in the Create data file are unchanged. The ghost file has the same Basis set, the same Fit set as for a normal atom. The values of the fit coefficients are irrelevant and could as well be put zero altogether: in the scf part of the create run on the ghost atom the fit coefficients will be set to zero after the first cycle since there is no charge density to be fitted.

Then the corresponding Create run is carried out.

```

$ADFBIN/adf -n1 << eor
Create Gh.C q=0 m=0 file=in.ghost
end input
eor

```

```
mv TAPE21 t21.C_ghost
```

The options 'q=' and 'm=' specify the nuclear charge and atomic mass respectively. Both are zero for a ghost atom: it is not a physical object, only the center for a set of functions.

In the same fashion the Oxygen and Chromium ghost atoms are created. The inputs for these are not shown here.

For the BSSE calculations we first do the 'normal' calculations of CO and Cr(CO)<sub>5</sub>, yielding the fragment files t21.CO and t21.CrCO5. The input files for these calculations are not shown here.

## BSSE for CO

For the CO BSSE calculation the standard CO must have been computed first. In the BSSE run a Cr(CO)<sub>5</sub> ghost fragment basis set is then added to the 'normal' CO input. The energy change (the printed 'bond energy' in the output) is the BSSE.

The input file for the CO-BSSE run is:

```
title BSSE for CO due to Cr(CO)5 ghost
noprnt sfo,frag,functions
```

atoms

```
Gh.Cr  0   0   0
Gh.C   -1.86  0   0
Gh.C    1.86  0   0
Gh.C    0   1.86  0
Gh.C    0  -1.86  0
Gh.C    0   0  -1.86
Gh.O    3.03  0   0
Gh.O   -3.03  0   0
Gh.O    0   3.03  0
Gh.O    0  -3.03  0
Gh.O    0   0  -3.03
  C     0   0   1.86   f=CO
  O     0   0   3.03   f=CO
```

end

fragments

```
Gh.Cr t21.Cr_ghost
Gh.C  t21.C_ghost
Gh.O  t21.O_ghost
CO   t21.CO
```

end

symmetry C(4V)

integration 4

endinput

In the output we find in the Bond Energy section:

	hartree	eV	kcal/mol	kJ/mol
	-----	-----	-----	-----
Pauli Repulsion				
Kinetic ( $\Delta T^0$ ):	0.000000000000025	0.0000	0.00	0.00
$\Delta V^{\text{Pauli}}$ Coulomb:	-0.000000000000021	0.0000	0.00	0.00
$\Delta V^{\text{Pauli}}$ LDA-XC:	-0.000000000000007	0.0000	0.00	0.00
	-----	-----	-----	-----
Total Pauli Repulsion:	-0.000000000000004	0.0000	0.00	0.00
(Total Pauli Repulsion = Delta E <sup>Pauli</sup> in BB paper)				
Steric Interaction				
Pauli Repulsion ( $\Delta E^{\text{Pauli}}$ ):	-0.000000000000004	0.0000	0.00	0.00
Electrostatic Interaction:	0.000000000000057	0.0000	0.00	0.00
(Electrostatic Interaction = Delta V <sub>elstat</sub> in the BB paper)				
	-----	-----	-----	-----
Total Steric Interaction:	0.000000000000054	0.0000	0.00	0.00

(Total Steric Interaction =  
Delta E<sup>0</sup> in the BB paper)

## Orbital Interactions

A1:	-0.001838637105191	-0.0500	-1.15	-4.83
A2:	0.000000000000000	0.0000	0.00	0.00
B1:	0.000000000000000	0.0000	0.00	0.00
B2:	0.000000000000000	0.0000	0.00	0.00
E1:	-0.002025936206895	-0.0551	-1.27	-5.32
-----				
Total Orbital Interactions:	-0.003864573312086	-0.1052	-2.43	-10.15

## Alternative Decomposition Orb.Int.

Kinetic:	-0.056036607909737	-1.5248	-35.16	-147.12
Coulomb:	0.048666200804427	1.3243	30.54	127.77
XC:	0.003505833793224	0.0954	2.20	9.20
-----				
Total Orbital Interactions:	-0.003864573312086	-0.1052	-2.43	-10.15
Residu (E=Steric+OrbInt+Res):	-0.000000000000002	0.0000	0.00	0.00
Total Bonding Energy:	-0.003864573312034	-0.1052	-2.43	-10.15

## Summary of Bonding Energy (energy terms are taken from the energy decomposition above)

```
=====
```

Electrostatic Energy:	0.000000000000057	0.0000	0.00	0.00
Kinetic Energy:	-0.056036607909712	-1.5248	-35.16	-147.12
Coulomb (Steric+OrbInt) Energy:	0.048666200804404	1.3243	30.54	127.77
XC Energy:	0.003505833793217	0.0954	2.20	9.20
-----				
Total Bonding Energy:	-0.003864573312034	-0.1052	-2.43	-10.15

The BSSE for CO is computed as 2.42 kcal/mole

**BSSE for Cr(CO)<sub>5</sub>**

In similar fashion the BSSE is computed for Cr(CO)<sub>5</sub>. In the BSSE run a ghost CO is added to the normal Cr(CO)<sub>5</sub> input:

```
title BSSE for Cr(CO)5 due to CO ghost
noprnt sfo,frag,functions
```

```
atoms
```

```
Cr 0 0 0 f=CrCO5
C 1.86 0 0 f=CrCO5
C -1.86 0 0 f=CrCO5
C 0 1.86 0 f=CrCO5
C 0 -1.86 0 f=CrCO5
C 0 0 -1.86 f=CrCO5
O 3.03 0 0 f=CrCO5
```

```

O -3.03 0 0 f=CrCO5
O 0 3.03 0 f=CrCO5
O 0 -3.03 0 f=CrCO5
O 0 0 -3.03 f=CrCO5
Gh.C 0 0 1.86
Gh.O 0 0 3.03
end

```

```

fragments
CrCO5 t21.CrCO5
Gh.C t21.C_ghost
Gh.O t21.O_ghost
end

```

```

symmetry C(4v)
integration 4

```

```

endinput

```

The Bond Energy result yields 1.93 kcal/mole for the BSSE.

## Bond Energy calculation with BSSE correction

The bonding of CO to Cr(CO)<sub>5</sub> is computed in the normal way (not included in the sample): from fragments CO and Cr(CO)<sub>5</sub>. The obtained value for the bond energy is then simply corrected for the two BSSE terms, 4.35 kcal/mole together.

## Relevance of Core Functions

The whole procedure explained above is repeated with now the Core Functions (the functions in the valence basis set that serve only for core-orthogonalization, for instance the 1s 5.40 in the Carbon basis set) removed from the Create data files used for the creation of the ghost atoms.

This yields as BSSE values for CO and Cr(CO)<sub>5</sub> respectively 2.33 and 1.88 kcal/mole (compare 2.42 and 1.93 kcal/mole for the case with Core Functions included). The net total effect of including/removing the Core Functions is therefore  $(2.42-2.33)+(1.93-1.88)=0.14$  kcal/mole. This is an order of magnitude smaller than the BSSE effect itself.

In the last calculation a PRINT instruction is inserted in the input file to let the program output the symmetry group representations, character table and multiplication table. This information is printed after the lists of basis and fit sets.

## BSSE and the size of the basis set

BSSE effects should diminish with larger bases and disappear in the limit of a perfect basis. This can be studied by comparing the BSSE for basis DZ, see above, with the BSSE for basis TZP. The procedure is completely similar to the one above and yields:

For the BSSE terms, using basis sets with Core Functions included: 0.7 kcal/mole for CO (compare: 2.4 kcal/mole for basis DZ), and 0.6 kcal/mole for Cr(CO)<sub>5</sub> (1.9 for basis DZ)

Without Core Functions the numbers are similar.

The total BSSE drops from 4.3 kcal/mole in basis DZ to 1.3 in basis TZP (if Core Functions are included in the Create runs for the ghosts), and changes very slightly when the Core Functions are omitted.

Create runs for the ghosts), and changes very slightly when the Core Functions are omitted.

A systematic study with *adf* of the BSSE in metal-carbonyl complexes can be found in Rosa, A., et al., Basis Set Effects in Density Functional Calculations on the Metal-Ligand and Metal-Metal Bonds of Cr(CO)<sub>5</sub>-CO and (CO)<sub>5</sub>. *Journal of Physical Chemistry*, 1996, 100: p. 5690-5696

## $N_2^+$ : Localized Hole

Sample directory: adf/e\_ModStPot\_N2+/

This calculation illustrates:

- a) How to specify the net total charge on a molecule
- b) How to enforce breaking the symmetry that is present in the start-up situation, in this case to localize a hole in the electron density on one of the two equivalent atoms.
- c) How to prevent the scf from oscillating back and forth between the two equivalent situations or from even restoring the unwanted symmetry

```
$ADFBIN/adf <<eor
title N2+ hole localization

atoms
N 0 0 -2.0
N 0 0 2.0
end

Basis
Type DZP
Core Small
End

symmetry C(lin) ! allow symmetry breaking

unrestricted

Occupations keeporbitals=3 &
! keeporbitals: let the density relax a bit, then fix the MO occupies
sigma 3 // 1 0 1
pi 2 // 2
end

CHARGE 1 1 ! this duplicates info from "OCCUPATIONS" (check)

modifystartpotential ! to break the symmetry in the start-up potential
N/1 0.5 0.5
N/2 4 1
end

end input
eor
```

The purpose of this run is to compute the  $N_2^+$  ion, with the hole localized on one of the atoms. In a very small system like  $N_2^+$  this is a tricky thing to do. The program has a tendency towards the symmetric solution, with the hole delocalized. A few trial runs, just putting a net +1 charge into the system, will reveal that clearly.

To achieve the desired situation we apply the key `modifystartpotential` to break the symmetry of the initial potential. A potential is generated as if the electronic cloud in the second N fragment is spin-polarized in a ratio 4:1 (this precise value is not very relevant), which achieves that *initially* a non-symmetric solution is obtained. The symmetry must be specified, lest the program determine and use the higher symmetry from the nuclear frame. This would prevent any symmetry breaking altogether.

Next, in order to prevent that the system relaxes to the symmetric situation, we apply the `keeporbitals` option of the occupations key. This fixes the occupied orbitals in the sense that in each scf cycle the program will try to keep the electrons in orbitals that resemble the previously occupied orbitals as much as possible.

The key `modifystartpotential` here demonstrated has a more relevant and less unstable application in larger systems. See the User's Guide for references.

## NNO: Core-electron binding energies

*Sample directory:* adf/e\_CEBE\_NNO/

ADF is well suited for calculating Core Electron Binding Energies (CEBEs). In this example it is shown how one can differentiate between the 1s CEBEs of the two non-equivalent nitrogen atoms in N<sub>2</sub>O, using a delta-SCF technique. It starts with a regular calculation that has the purpose of preparing a reference TAPE21 file for the NNO molecule, which will later be useful in the energy analysis. The result file is saved to t21.NNO.

The same GGA functional is specified throughout the run. The amount of output is reduced by using some print keys.

```
$ADFBIN/adf -n1 << eor
create N $ADFRESOURCES/TZ2P/N
xc
gradients pw86x pw91c
end
end input
eor
```

```
mv TAPE21 t21.N
```

```
$ADFBIN/adf -n1 << eor
create O $ADFRESOURCES/TZ2P/O
xc
gradients pw86x pw91c
end
end input
eor
```

```
mv TAPE21 t21.O
```

The prepare the nitrogen atom with a core hole (restricted) will be used as a fragment later. This enables selection of where the core hole should be.

```
$ADFBIN/adf -n1 << eor
title N atom core hole
```

```
ATOMS
N 0.0 0.0 0.0
end
```

```
fragments
N t21.N
end
```

```
xc
gradients pw86x pw91c
end
```

```
eprint
```

```
SFO noeig noovl
end
```

```
occupations
s 1 2
p 3
end
```

```
end input
eor
```

```
mv TAPE21 t21.N_ch
```

Now perform the restricted ground state molecule for analysis later. The TAPE21 result file is saved.

```
$ADFBIN/adf << eor
title NNO
```

```
noprint sfofragpop fragsfo
```

```
xc
gradients pw86x pw91c
end
```

```
ATOMS
N 0.0 0.0 -1.1284
N 0.0 0.0 0.0
O 0.0 0.0 1.1841
end
```

```
fragments
N t21.N
O t21.O
end
```

```
eprint
SFO noeig noovl
end
```

```
end input
eor
```

```
mv TAPE21 t21.NNO
```

Next follow two sets of almost identical calculations in which a 1s electron is removed from one or the other N atom (please note that the deepest s level is associated with the 1s of the oxygen atom). The molecular NNO result file is used as fragment. An unrestricted calculation is done and a positive charge is specified. The final result file for the molecule with the core hole is saved. Then another calculation is done to conveniently obtain the energy with respect to the normal molecule. This is repeated for a core hole on the other N atom.

```
$ADFBIN/adf <<eor
title NNO unrestricted core hole
```

```
noprint sfofragpop fragsfo
```

```
ATOMS
```

```
N 0.0 0.0 -1.1284 f=N_ch
N 0.0 0.0 0.0 f=N
O 0.0 0.0 1.1841 f=O
```

```
end
```

```
xc
```

```
gradients pw86x pw91c
end
```

```
fragments
```

```
N_ch t21.N_ch
N t21.N
O t21.O
```

```
end
```

```
eprint
```

```
SFO noeig noovl
end
```

```
unrestricted
```

```
charge 1 1
```

```
occupation
```

```
sigma 1 1 1 4 // 1 0 1 4
pi 4 // 4
```

```
end
```

```
end input
```

```
eor
```

```
mv TAPE21 t21.NNO.unr1
```

In the second calculation the result file of one of the unrestricted NNO calculations is used as restart file, which ensures that the hole stays at its place, because the starting density is already correct. The result file t21.NNO for the normal NNO calculation is specified as fragment to serve as an energy reference. The final Bonding Energy printed by ADF indicates what the CEBE is. However, please check Refs.[Chong, 2002 #1239][Chong, 2002 #1238] for more detailed information on Core-Electron Binding Energies. These references also contain information on empirical corrections that may have to be made on the final numbers.

```
$ADFBIN/adf <<eor
```

```
title NNO unr. core hole
```

```
noprint sfofragpop fragsfo
```

```
xc
```

```
gradients pw86x pw91c
end
```

```
restart t21.NNO.unr1
```

```
ATOMS
```

```
N 0.0 0.0 -1.1284 f=NNO
```

```
N 0.0 0.0 0.0 f=NNO
O 0.0 0.0 1.1841 f=NNO
end
```

```
eprint
SFO noeig noovl
end
```

```
fragments
NNO t21.NNO
end
```

```
unrestricted
charge 1 1
```

```
occupation
sigma 1 1 1 4 // 1 0 1 4
pi 4 // 4
end
```

```
end input
eor
```

Similarly, one could easily have prepared an oxygen with a core hole and determined the CEBE of the oxygen 1s atom.

## HCl: Solvent Effects

*Sample directory:* adf/e\_Solv\_HCl/

Computing solvent effects, with the COSMO model, is illustrated in the HCl example.

After a non-solvent (reference) calculation, which is omitted here, two solvent runs are presented, with somewhat different settings for a few input parameters. The block key Solvation controls all solvent-related input.

All subkeys in the SOLVATION block are discussed in the User's Guide. Most of them are rather technical and should not severely affect the outcome. Physically relevant is the specification of the solute properties, by the SOLVENT subkey: the dielectric constant and the effective radius of the solvent molecule.

A rather strong impact on the computation times has the method of treating the "C-matrix". There are 3 options (see the User's Guide): EXACT is the most expensive, but presumably most accurate. POTENTIAL is the cheapest alternative and is usually quite adequate. EXACT uses the exact charge density for the Coulomb interaction between the molecular charge distribution and the point charges (on the Van der Waals type molecular surface) which model the effects of the solvent. The alternatives, notably "POTENTIAL", use the *fitted* charge density instead. Assuming that the fit is a fairly accurate approximation to the exact charge density, the difference in outcome should be marginal.

```
$ADFBIN/adf << eor
TITLE HCl(1) Solv-excl surfac; Gauss-Seidel (old std options)
```

```
SYMMETRY NOSYM
```

```
ATOMS Cartesian
```

```
  H  0.000000  0.000000  0.000000  R=1.18
  Cl 1.304188  0.000000  0.000000  R=1.75
END
```

```
Fragments
```

```
  H t21.H
  Cl t21.Cl
End
```

```
SOLVATION
```

```
  Solvent      epsilon=78.8 radius=1.4
  SurfaceType  esurf
  DivisionLevel ND=4 min=0.5 Ofac=0.8
  ChargeUpdate Method=Gauss-Seidel
  DiscAttributes SScale=0.01 LEGendre=10 TOLerance=1.0d-2
  SCF          Variational
  C-Matrix     Exact
END
```

```
NOPRINT Bas EigSFO EKin SFO, frag, functions
```

```
EPRINT
SCF NoEigvec
END
END INPUT
eor
```

```
rm TAPE21 logfile
```

In the second solvent run, another (technical) method is used for determining the charge distribution on the cavity surface (conjugate-gradient versus Gauss-Seidel in the previous calculation), and the POTENTIAL variety is used for the C-matrix handling. The results show that it makes little difference in outcome, but quite a bit in computation times.

```
$ADFBIN/adf << eor
```

```
TITLE HCl(9) NoDisk and Cmatrix potential
```

```
FRAGMENTS
```

```
  H  t21.H
```

```
  Cl t21.Cl
```

```
END
```

```
ATOMS Cartesian
```

```
  H  0.000000  0.000000  0.000000  R=1.18
```

```
  Cl 1.304188  0.000000  0.000000  R=1.75
```

```
END
```

```
SOLVATION
```

```
  Solvent  epsilon=78.8 radius=1.4
```

```
  SurfaceType  esurf
```

```
  DivisionLevel ND=4 min=0.5 Ofac=0.8
```

```
  ChargeUpdate  Method=conjugate-gradient
```

```
  SCF  Variational
```

```
  C-Matrix  POTENTIAL
```

```
END
```

```
NOPRINT Bas EigSFO EKin SFO, frag, functions
```

```
EPRINT
```

```
SCF NoEigvec
```

```
END
```

```
END INPUT
```

```
eor
```

## **N<sub>2</sub> and PtCO: Electric Field, Point Charge(s), use of Basis keyword**

*Sample directories:* `adf/e_Efield.PntQ_N2/` and `adf/e_Field_PtCO`

Two illustrations of applying the very useful BASIS keyword and of application of an Electric Field.

For N<sub>2</sub>, three calculations are provided: 1) a normal N<sub>2</sub> run as a reference with the BASIS keyword, 2) with a homogeneous electric field, 3) with a point charge.

In this example, no Create run is needed in the input file, because the first molecular calculation uses the BASIS keyword. If the \$ADFBIN/adf script finds this keyword, it will first generate a new input file which will then be executed. The new input file will contain the required Create run for the N atom in this case. The proper xc functional and relativistic options will automatically be selected by the BASIS keyword. This includes Dirac calculations in case of relativistic runs. The output files is identical to what would have appeared if one would provide the Create runs explicitly in the input file. It also copies the atomic input, so that everything can be checked.

```
$ADFBIN/adf -n1 << eor
title N2 reference for comparison with E-Field runs
```

```
atoms
N 0 0 -.55
N 0 0 +.55
end
```

```
Basis
Type DZP
Core Small
End
```

```
end input
eor
```

```
rm TAPE21 logfile
```

```
$ADFBIN/adf << eor
scf
conv 1e-8
end
```

```
title N2 in a homogeneous electric field
```

```
atoms
N 0 0 -.55
N 0 0 +.55
end
fragments
N t21.N
end
```

```
EField 0 0 0.01
```

```
end input
eor

rm TAPE21 logfile

$ADFBIN/adf << eor
title N2 polarized by a point charge on the axis

EField
  0 0 3.0 1.0
end

atoms
  N 0 0 -.55
  N 0 0 .55
end

Fragments
  N t21.N
end

endinput
eor
```

In the second  $n_2$  run the homogeneous field is supplied with the key efield, used as simple key: one record, data on the same line as the keyword. The field strength is specified in atomic units.

Homogeneous electric fields can be used to study the polarizability: for sufficiently small fields the dipole moment should respond linearly.

For point charges, the third calculation, the block form of the key efield must be used. The program first tries to find data on the same line as the keyword (defining a homogeneous field). If this is absent, a data block is expected with point-charge specifications: x, y, z and q.

The coordinates are in the same units as in the atoms block (angstrom by default) (but always Cartesian). Q is the charge in elementary units (+1 for a proton).

Point charges can be used for instance to simulate crystal fields (Madelung potential).

Note: the symmetry will be determined automatically by the program as C(lin), rather than D(lin), in the two runs that involve an electric field: the fields break the symmetry.

For PtCO, a fairly large electric field is applied in combination with a tight SCF convergence criterion.

The BASIS keyword in this example illustrates how different choices can be made for different atoms (in this case a frozen core for Pt).

```
Basis
Type DZ
Core None
Pt Pt.4d
END
```

## CuH<sup>+</sup>: calculation of S<sup>2</sup>

*Sample directory:* adf/e\_CuH+\_S-squared/

Example calculates expectation value of S<sup>2</sup> ( $\langle S^2 \rangle$ ) of CuH<sup>+</sup> in various symmetries, using unrestricted density functional theory. Last example in this example file calculates this value in the case there are more beta electrons than alpha electrons.

```
$ADFBIN/adf << eor
```

```
Title calculate expectation value of S-squared
```

```
ATOMS Z-Matrix
```

```
Cu 0 0 0
```

```
H 1 0 0 1.463
```

```
END
```

```
CHARGE 1.0 -1.0
```

```
Unrestricted
```

```
FRAGMENTS
```

```
H t21.H
```

```
Cu t21.Cu
```

```
END
```

```
endinput
```

```
eor
```

## Time-dependent DFT applications

### Au<sub>2</sub>: Response Properties

*Sample directory:* adf/e\_Au2\_Resp/

A calculation of response properties of the Au<sub>2</sub> dimer, with ZORA relativistic corrections

```
$ADFBIN/adf << eor  
Title Au2, Response Properties
```

```
XC  
  GGA LB94  
End
```

```
Relativistic Scalar ZORA  
CorePotentials t12.Au
```

```
Atoms  
  Au  0.0 0.0 1.236  
  Au  0.0 0.0 -1.236  
End
```

```
Fragments  
  Au  t1.Au  
End
```

```
Symmetry D(8h)
```

```
Excitations  
  Lowest 10  
  TOLERANCE 1d-10  
End
```

```
Response  
  AllComponents  
End
```

```
End Input  
eor
```

In the response module infinite symmetries cannot be handled (see the User's Guide), so we specify a lower subgroup in the input file, here D(8h).

In this sample run the LB94 potential is used. The implementation implies that the XC potential is evaluated from the exact charge density, rather than the cheaper and faster fitted density (as is the case for other XC functionals). This means that the computation times are longer. In a small molecule like Au<sub>2</sub> this hardly shows, but in larger molecules the differences may be more significant.

Excitation energies are computed, in principle the lowest 10 in each irrep of the symmetry (see, however, the User's Guide).

## Hyperpol: Hyperpolarizabilities of He and H<sub>2</sub>

Sample directory: adf/e\_Hyperpol/

This sample illustrates the computation of (hyper) polarizability tensors for the He atom and the H<sub>2</sub> molecule.

The symmetry is specified, because the Response module in ADF cannot yet handle the infinite symmetries ATOM, C(lin), D(lin).

```
$ADFBIN/adf <<EOR
Title expt geometrie H2(VII),VWN
noprnt sfo,frag,functions
```

```
Symmetry C(8v)
```

```
Atoms
  H 0 0 -0.37305
  H 0 0  0.37305
End
```

```
Fragments
  H t21.H7
End
```

```
Response
  HyperPol 0.03
  DynaHyp
  AllComponents
End
```

```
EField 0 0 0.001
```

```
end input
EOR
```

The Response data block specifies (AllComponents) that not only the (default) zz-dipole polarizability is to be computed, but the complete tensor. The subkey HyperPol instructs the program to compute *hyperpolarizabilities* and not only polarizabilities. The DynaHyp subkey implies that the *frequency-dependent* (hyper)polarizability is calculated. In that case the main laser frequency has to be specified, in hartree units, after the HyperPol subkey.

Only the first hyperpolarizability has been implemented in ADF. Some information on second hyperpolarizabilities can be obtained from the calculation of the first one in a finite field (EFIELD).

In similar fashion the frequency-dependent hyperpolarizability is computed for He, but only the zzz-component because now the AllComponents subkey is omitted.

```
$ADFBIN/adf <<EOR
Title hyperpolarizability He with the LB94 potential
noprnt sfo,frag,functions
```

```
Atoms
```

```
He 0 0 0
End

XC
GGA LB94
END

Fragments
He t21.He8
End

Response
HyperPol 0.07
DynaHyp
End

integration 5.0

EField 0 0 0.001

Symmetry C(8v)

end input
EOR
```

## HF: Dispersion Coefficients

*Sample directory:* adf/e\_Disper\_HF/

General dispersion coefficients (beyond de dipole-dipole  $C_6$  interaction coefficient) are computed with the auxiliary program DISPER. It uses two output files from previous ADF Response calculations. In the example, the two ADF runs are one and the same and the relevant TENSOR output file is used twice.

```
$ADFBIN/adf <<EOR
```

```
title Van der Waals coefficients HF
```

```
atoms
```

```
H 0 0 -0.8708056087
```

```
F 0 0 0.04619439132
```

```
end
```

```
Basis
```

```
Type DZP
```

```
Core Small
```

```
End
```

```
symmetry C(8v)
```

```
RESPONSE
```

```
MAXWAALS 8
```

```
VANDERWAALS 7
```

```
ALLTENSOR
```

```
ALLCOMPONENTS
```

```
END
```

```
end input
```

```
EOR
```

Polarizabilities are computed at 7 (imaginary) frequencies between 0 and infinity. The program determines internally the actual frequency *values* in this range to use. The user only specifies the number of them, thereby determining the precision of, in fact, a numerical integration over the zero-infinity frequency range. A value of 7 is rather low.

MaxWaals determines that not only the  $C_6$  but also  $C_7$  and  $C_8$  coefficients are computed. A value higher than 8 would not be recommended, because the available basis sets would be inadequate for higher coefficients.

In DISPER calculations the preparatory Response calculation *must* use the AllTensor and AllComponents subkeys.

The calculation produces a file TENSOR. The subsequent DISPER run uses two such files. In this example, both are taken from the same ADF run, copying the TENSOR file to, respectively, tensorA and tensorB. These names are prescribed for a DISPER calculation.

```
cp TENSOR tensorA
```

```
cp TENSOR tensorB
```

```
$ADFBIN/disper -n1 << eor
```

eor

The DISPER program needs no other input than just the files tensorA and tensorB, which must both be present as local files. Note the '-n1' flag: this enforces that a single-node (non-parallel) run is performed. The current implementation does not support parallelization of DISPER, because the kid processes may not have the (local to the master!) files tensorA and tensorB.

## DMO: Circular Dichroism spectrum

*Sample directory:* adf/e\_DMO\_CD/

If the subkey CDSPECTRUM is included in the key EXCITATIONS, the rotatory strength is calculated for the calculated excitations, in order to calculate the CD (Circular Dichroism) spectrum. Only useful for chiral molecules.

With the VELOCITY keyword also the dipole-velocity representation of the rotatory strength is calculated.

Note: results will be physically meaningless due to small basis set. purpose of this job is to provide a test case for the CD implementation

Do not use less strict convergence criteria than default, better to use tighter criteria. The approximations in the evaluation of the integrals one makes with the linear scaling techniques are effectively switched off by setting LINEARSCALING 100 (recommended to use this).

Usage:

```
$ADFBIN/adf <<eor
```

```
TITLE dimethyloxirane excitations + CD
```

```
COMMENT
```

```
results will be physically meaningless due to small basis set.  
purpose of this job is to provide a test case for the CD implementation  
END
```

```
XC
```

```
gga becke perdew  
END
```

```
Basis
```

```
Type DZP  
Core Small  
End
```

```
ATOMS
```

```
O 0.000129 1.141417 0.000023  
C -0.597040 -0.094320 0.428262  
C 0.596952 -0.094328 -0.428223  
H -0.442927 -0.302650 1.487698  
H 0.442944 -0.302474 -1.487720  
C -1.978779 -0.386617 -0.093924  
H -2.723275 0.220579 0.429114  
H -2.043506 -0.157697 -1.159810  
H -2.236045 -1.439970 0.055144  
C 1.978716 -0.386693 0.093893  
H 2.236030 -1.439985 -0.055498  
H 2.723156 0.220701 -0.429005  
H 2.043497 -0.158088 1.159845  
END
```

```
linearscaling 100
```

```
excitations  
  cdspectrum  
  onlysinglet  
  velocity  
  lowest 10  
end
```

```
END INPUT  
eor
```

## DMO: Optical Rotation Dispersion

Sample directory: adf/e\_DMO\_ORD/

If the subkey OPTICALROTATION is included in the key RESPONSE, the (frequency dependent) optical rotation is calculated.

Note: results will be physically meaningless due to small basis set. purpose of this job is to provide a test case for the ORD implementation

Do not use less strict convergence criteria than default, better to use tighter criteria. The approximations in the evaluation of the integrals one makes with the linear scaling techniques are effectively switched off by setting LINEARSCALING 100 (recommended to use this).

Usage:

```
$ADFBIN/adf <<eor
```

```
TITLE dimethyloxirane excitations + ORD
```

```
COMMENT
```

```
  results will be physically meaningless due to small basis set.
```

```
  purpose of this job is to provide a test case for the ORD implementation
```

```
END
```

```
XC
```

```
  gga becke perdew
```

```
END
```

```
Basis
```

```
  Type DZP
```

```
  Core Small
```

```
End
```

```
ATOMS
```

```
O      0.000129  1.141417  0.000023
```

```
C     -0.597040 -0.094320  0.428262
```

```
C      0.596952 -0.094328 -0.428223
```

```
H     -0.442927 -0.302650  1.487698
```

```
H      0.442944 -0.302474 -1.487720
```

```
C     -1.978779 -0.386617 -0.093924
```

```
H     -2.723275  0.220579  0.429114
```

```
H     -2.043506 -0.157697 -1.159810
```

```
H     -2.236045 -1.439970  0.055144
```

```
C      1.978716 -0.386693  0.093893
```

```
H      2.236030 -1.439985 -0.055498
```

```
H      2.723156  0.220701 -0.429005
```

```
H      2.043497 -0.158088  1.159845
```

```
END
```

```
linearscaling 100
```

```
response
```

```
allcomponents
```

```
opticalrotation
```

```
end  
END INPUT  
eor
```

## Special exchange-correlation functionals

### CO: asymptotically correct xc potentials

Sample directory adf/e\_CO\_model

For property calculations, xc potentials with asymptotically correct  $(-1/r)$  behavior outside the molecule, the results tend to be superior to regular LDA or GGA calculations. This is especially true for small molecules and for properties that depend heavily on the proper description of the outer region of the molecule. In the example, all-electron basis sets are used. This is mandatory for the SAOP potential.

```
$ADFBIN/adf -n1 <<EOR
create C $ADFRESOURCES/TZ2P/C
end input
EOR
mv TAPE21 t21.C
```

```
$ADFBIN/adf -n1 <<EOR
create O $ADFRESOURCES/TZ2P/O
end input
EOR
mv TAPE21 t21.O
```

In the next example, excitation energies are calculated with the GRACLB potential. This potential requires one number as argument: the experimental ionization potential in atomic units. This number can be either based on an experimental value, or on previous GGA total energy calculations.

```
$ADFBIN/adf <<EOR
title CO excitations grac potential
```

```
INTEGRATION 6.0
```

```
XC
  Model GRACLB 0.515
End
```

```
Atoms
O 0      0 0
C 1.128205364 0 0
end
```

```
Excitation
  Lowest 10
  Onlysing
End
```

```
Fragments
  O t21.O
  C t21.C
End
```

```
end input  
EOR
```

```
rm TAPE21 logfile
```

The same calculation with the SAOP xc potential would differ in the XC block only:

```
XC  
Model SAOP  
End
```

SAOP depends on the orbitals which makes it more expensive to evaluate than GRAC for large molecules.

## OH: Meta-GGA energy functionals

*Sample directory adf/e\_OH\_MetaGGA*

A large even-tempered basis set calculation of the atomization energy of OH using various modern GGA and meta-GGA post-SCF energy expressions.

In the Create runs, a large even-tempered basis set is selected for O and H, which should give results closer to the basis set limit than the regular ADF basis sets. For both atoms, a second atomic calculation follows the Create run, in order to enable a comparison to the true atoms, rather than the artificial spherically symmetric atom from the Create run. This is achieved by specifying the keywords

```
unrestricted
charge 0 2
symmetry C(lin)
occupations
sigma 3 // 3
pi 2 // 0
end
```

in the case of oxygen. This fixes the proper occupations. The result files of both the Create runs and the atomic correction runs are stored.

In the molecular calculation, the symmetry of the molecule is explicitly broken and the occupations are specified in order to avoid the fractional occupations that ADF would otherwise choose. Although it is not said that such a solution would be inferior, the integer occupation solution is the one which allows direct comparison to literature results obtained with other programs.

One of the new GGA potentials has been specified for the xc potential and the keyword METAGGA implies that a series of GGA and meta-GGA xc energies is to be calculated and compared to those energies from the atomic calculations.

```
METAGGA
symmetry C(lin)
xc
GGA PBE
end
```

A fairly high numerical integration has been specified. For meta-GGA calculations we do recommend this, at least for the time being, as the numerical stability of the results tends to be somewhat lower than for regular GGA calculations.

The block key ENERGYFRAG

```
ENERGYFRAG
O t21.unr.O
H t21.unr.H
END
```

implies that the meta-GGA result must not only be compared to the spherically symmetric results from the Create runs, but also to the non-spherical atoms.

The molecular output file prints the PBE Total Bonding energy as usual (in different energy units).

Then a prints a list of 'Total Bonding Energies' for many different Exc functionals, including PBE. Because

the numerical approach to obtain the two PBE results is somewhat different, small differences may occur between the two numbers. You now have an overview of the bonding energies of all (meta)GGA functionals currently implemented in ADF. This should give a good indication of the theoretical error bar or the uncertainty in the xc approximation.

Total Bonding Energy:            -0.286127457276205        -7.7859            -179.55            -751.23

#### TOTAL BONDING ENERGIES FROM VARIOUS XC FUNCTIONALS

with respect to fragments in FRAGMENTS input block

	hartree	eV	kcal/mol	kJ/mol
Total Bonding Energy with respect to FRAGMENTS				
XC Energy Functional				
=====				
FR: KCIS-modified [1] =	-0.2755742057	-7.4987587362	-172.9254430523	-723.5200549587
FR: KCIS-original [2] =	-0.2777894828	-7.5590395194	-174.3155506035	-729.3362649626
FR: PKZB [3] =	-0.2815570432	-7.6615600946	-176.6797306630	-739.2279943483
FR: VS98 [4] =	-0.3017049511	-8.2098127875	-189.3227350810	-792.1263249228
FR: LDA(VWN) [5] =	-0.2887564297	-7.8574654492	-181.1974143810	-758.1299830563
FR: PW91 [6] =	-0.2876922977	-7.8285089331	-180.5296614163	-755.3361046473
FR: BLYP [7] =	-0.2770745036	-7.5395839361	-173.8668943006	-727.4590869882
FR: BP [8] =	-0.2855241909	-7.7695117221	-179.1691537365	-749.6437405057
FR: PBE [9] =	-0.2858734106	-7.7790144775	-179.3882924288	-750.5606167957
.....				

The same energy comparison is done with respect to the fragments (which most currently be atomic) in the ENERGYFRAG block. These are the numbers which should be comparable to experimental numbers.

Finally, the references for the various Exc functionals are printed in the output file.

XC Energy Functional				
=====				
EF: KCIS-modified [1] =	-0.1713622482	-4.6630059333	-107.5314455812	-449.9115690750
EF: KCIS-original [2] =	-0.1701706820	-4.6305817515	-106.7837263654	-446.7831118709
EF: PKZB [3] =	-0.1716508948	-4.6708604097	-107.7125740668	-450.6694106602
EF: VS98 [4] =	-0.1712676117	-4.6604307410	-107.4720602503	-449.6631008503
EF: LDA(VWN) [5] =	-0.1980694328	-5.3897456994	-124.2904587006	-520.0312800855
EF: PW91 [6] =	-0.1759694023	-4.7883730257	-110.4224787188	-462.0076517434
EF: BLYP [7] =	-0.1748768123	-4.7586421272	-109.7368680765	-459.1390568111
EF: BP [8] =	-0.1785853781	-4.8595573769	-112.0640284617	-468.8758958794
EF: PBE [9] =	-0.1751227104	-4.7653333576	-109.8911714787	-459.7846622469
.....				

Similar calculations can be done to obtain energy differences between different molecules. In that case the ENERGYFRAG keyword is not operational though. No detailed breakdown of the bonding energy is currently available for these new energy functionals. Experience shows that the energy values depend only mildly on the chosen xc functional for the xc potential.

## H: SIC-VWN potential

*Sample directories:* adf/e\_H\_SICVWN/

Computation of the hydrogen atom with the SIC-VWN potential, should give the exact result ( $E=-0.5$  a.u.).

Note: adf with the SIC-VWN only runs correctly serial, and symmetry NOSYM is required.

```
$ADFBIN/adf -n1 << eor
TITLE H atom, SIC-VWN (should be exact)
SYMMETRY NOSYM
UNRESTRICTED
CHARGE 0 1
ATOMS
  1 H 0.0000 0.0000 0.0000
END
INTEGRATION 6.0 6.0
FRAGMENTS
  H t21.H
END
XC
  LDA VWN
END
SICOEP
  IPRINT 1
  SELF 35
END
DEPENDENCY fit=1e-10 bas=1e-8
SINGULARFIT FRUGAL
END INPUT
eor
```

## QM/MM calculations

### QMMM\_Butane: Basic QMMM Illustration

Sample directory: adf/e\_QMMM\_Butane/

This example is a simple illustration of the QMMM functionality: half of the butane molecule is treated quantum-mechanically, the other half by molecular mechanics.

```
$ADFBIN/adf << eor
Title BUTANE in Z-matrix input
```

(Omitted in this printout: the usual specifications of fragments, symmetry, integration accuracy, -)

QMMM

```
FORCEFIELD_FILE $ADFRESOURCES/ForceFields/amber95.ff
RESTART_FILE mm.restart
OUTPUT_LEVEL=2
WARNING_LEVEL=2
ELSTAT_COUPLING_MODEL=0
```

LINKS

```
1 - 4 1.38000 H
```

SUBEND

MM\_CONNECTION\_TABLE

```
1 CT QM 2 3 4 5
```

```
2 HC QM 1
```

```
3 HC QM 1
```

```
4 CT LI 1 9 13 14
```

```
5 CT QM 1 6 7 8
```

```
6 HC QM 5
```

```
7 HC QM 5
```

```
8 HC QM 5
```

```
9 CT MM 4 10 11 12
```

```
10 HC MM 9
```

```
11 HC MM 9
```

```
12 HC MM 9
```

```
13 HC MM 4
```

```
14 HC MM 4
```

SUBEND

End

Atoms Internal

C	0	0	0	0	0	0
H	1	0	0	B1	0	0
H	1	2	0	B2	A1	0
C	1	2	3	B3	A2	D1
C	1	2	3	B4	A3	D2
H	5	1	2	B5	A4	D3
H	5	1	6	B6	A5	D4

```
H 5 1 6      B7      A6      D5
C 4 1 2      B8      A7      D6
H 9 4 1      B9      A8      D7
H 9 4 10     B10     A9      D8
H 9 4 10     B11     A10     D9
H 4 1 9      B12     A11     D10
H 4 1 9      B13     A12     D11
End
```

GeoVar

....

In the QMMM key block, the MM connection table identifies the atoms as belonging to either the QM (quantum mechanics) part, or the MM (molecular mechanics) part, or to the set of LI (link) atoms, which define the connection between the QM and the MM regions. Order and numbering are one-to-one with the list under the Atoms key.

The Link atom, part of the MM section of the system, is associated with a *capping atom*, in the QM part of the system. The Links subkey block specifies for each LI atom defined under the MM\_Connection\_Table subkey block the chemical type of the replacing capping atom (here: H). On the same line we find the ratio of the QM atom – LI atom distance to the QM atom – capping atom distance (here: 1.38), and the numbers (1 and 4) of the involved QM atom and LI atom.

The other subkeys in the QM key block are simple subkeys. They specify the file with the force field parameters to be used in the MM subsystem, the (restart) file to write MM data to, print and warning levels and a code for the electrostatic coupling model to use. See the QMMM manual for a detailed discussion of all options.

The calculation is a simple geometry optimization (the Geometry key is not displayed here, but is contained in the full input). This consists of a repeated two-step process. At the first step, the MM system is kept frozen, the SCF equations are solved for the QM system, where potentials resulting from the MM system are included, and gradients on the QM atoms are computed from the SCF solution. At the second step, the QM system's geometry is updated and then kept frozen while the MM system's geometry is optimized (converged) for that particular QM configuration. And so on, until the whole combined system is self-consistently converged.

## QMMM\_CYT

*Sample directory:* adf/e\_QMMM\_CYT/

See the QMMM manual , where this case is used as a 'walk through' for the QMMM feature.

It is a more or less straightforward application of QMMM to geometry optimization (Cytocine). In the Atoms block all atoms are listed (QM as well as MM). All QMMM aspects, such as which atoms belong to the QM "core" and which are to be treated by the approximate MM method, are found in the QMMM key block, and its various subkey blocks. The remainder of the input file is not different from what it would be in a non-QMMM run.

The standard amber95 force field is used, which is located in the database of the ADF distribution.

```
$ADFBIN/adf << eor
```

```
Title CYT amber95 - Cartesian Geometry Optimization
```

```
Fragments
```

```
  C t21.C
```

```
  H t21.H
```

```
End
```

```
Charge 0 0
```

```
Atoms Cartesian
```

```
 1 C    1.94807  3.58290 -0.58162
 2 C    1.94191  3.61595  1.09448
 3 H    1.69949  4.49893 -1.05273
 4 H    2.99455  3.17964 -0.86304
 5 C    0.94659  2.40054 -0.92364
 6 N   -1.74397 -3.46417  0.31178
 7 C   -1.00720 -2.20758  0.33536
 8 C   -1.66928 -1.00652  0.31001
 9 C   -0.92847  0.25653  0.34895
10 N    0.43971  0.26735  0.38232
11 N    0.36409 -2.20477  0.28992
12 C    1.09714 -0.95413  0.22469
13 H   -2.89781 -3.50815  0.31746
14 H   -1.21484 -4.49217  0.31721
15 H   -2.80940 -0.93497  0.30550
16 H   -1.55324  1.21497  0.33885
17 C    1.23309  1.44017  0.30994
18 O    2.58277 -1.01636  0.23914
19 H    2.37276  1.25557  0.29984
20 O    1.02358  2.43085  1.50880
21 H    1.17136  1.95097 -1.87367
22 H   -0.10600  2.77333 -0.80348
23 H    1.62170  4.54039  1.51392
24 H    2.99608  3.28749  1.41345
```

```
End
```

## QMMM

```
FORCEFIELD_FILE $ADFRESOURCES/ForceFields/amber95.ff
RESTART_FILE mm.restart
OUTPUT_LEVEL=1
WARNING_LEVEL=2
ELSTAT_COUPLING_MODEL=1
```

## LINK\_BONDS

```
1 - 5 1.38000 H
1 - 2 1.38030 H
```

## SUBEND

## MM\_CONNECTION\_TABLE

```
1 CT QM 2 3 4 5
2 CT LI 1 20 23 24
3 HC QM 1
4 HC QM 1
5 CT LI 1 17 21 22
6 N2 MM 7 13 14
7 CA MM 6 8 11
8 CM MM 7 9 15
9 CM MM 8 10 16
10 N* MM 9 12 17
11 NC MM 7 12
12 C MM 10 11 18
13 H MM 6
14 H MM 6
15 HA MM 8
16 H4 MM 9
17 CT MM 5 10 19 20
18 O MM 12
19 H2 MM 17
20 OS MM 2 17
21 HC MM 5
22 HC MM 5
23 H1 MM 2
24 H1 MM 2
```

## SUBEND

## CHARGES

```
1 0.0 CT
2 0.0 CT
3 0.0 HC
4 0.0 HC
5 0.0 CT
6 -0.9530 N2
7 0.8185 CA
8 -0.5215 CM
9 0.0053 CM
10 -0.0484 N*
11 -0.7584 NC
12 0.7538 C
13 0.4234 H
14 0.4234 H
```

```
15 0.1928 HA
16 0.1958 H4
17 0.0066 CT
18 -0.6252 O
19 0.2902 H2
20 -0.2033 OS
21 0.0000 HC
22 0.0000 HC
23 0.0000 H1
24 0.0000 H1
SUBEND
```

END

Geometry

```
Iterations 20
Converge E=1.0E-3 Grad=0.0005
Step Rad=0.3 Angle=5.0
End
```

XC

```
LDA VWN
GGA PostSCF Becke Perdew
End
```

Integration 3.0

SCF

```
Iterations 60
Converge 1.0E-06 1.0E-6
Mixing 0.20
DIIS N=10 OK=0.500 CX=5.00 CXX=25.00 BFAC=0.00
End
```

End Input

eor

## QMMM\_Surface: Ziegler-Natta catalysis

*Sample directory:* adf/e\_QMMM\_Surface/

This is an example of a Ziegler-Natta type catalytic system: a TiCl complex embedded in a MgCl surface with two organic substrates also attached to the surface. To make the computation faster, the QMMM approach is applied. The QM part includes only the active site and a piece of the MgCl surface.

The computation is formally a geometry optimization, but to keep the sample doable in a reasonable time the sample performs only one geometry update step. In the optimization, all of the MgCl surface atoms are frozen.

The standard force field has been modified to accommodate this calculation. The modified force field file is part of the sample run script. In this modified file, bonds are defined between Mg-Cl atoms in the MM connection table. This results in some torsions where the atoms are collinear. To rectify this problem, the torsional potentials for these atoms are set to potential type '0' (no potential).

There are no capping atoms mediating the bonds between the QM and MM regions because the boundary goes through the MgCl surface, which is ionically bound.

```
cat << eor > champ_de_force.ff
YBYL/TRIPOS FORCE FIELD FILE FOR ADF QM/MM
MODIFIED WITH UFF1.01 FOR Si Mg Ti Cl
L. Petitjean 15.11.1999
*****
```

(Most of the contents of the modified force field file is omitted here. You quickly get the difference with the standard sybyl force field file in the ADF database by running a UNIX *diff* on the two files.

```
=====
eor
```

```
$ADFBIN/adf << eor
Title  ADF-QMMM in a surface study
NoPrint SFO, Frag, Functions

MaxMemoryUsage 20

! keywords for calculation methods and optimization
XC
LDA  VWN
GGA  BLYP
End

Geometry
Optim      Cartesian Selected
Iterations 1
HessUpd    BFGS
Converge   e=1e-4 grad=1e-3 rad=1e-2
Step       rad=0.15
END
```

The 'Iterations 1' subkey specification in the Geometry block specifies that only one step in the optimization is carried out.

Integration 3.0 3.0

SCF

Iterations 250

Converge 1E-6 1E-6

Mixing 0.2

DIIS N=10 OK=0.5 cyc=5 CX=5.0 BFAC=0

End

! keywords for molecule specification

Charge 0 0

Atoms Cartesian

1 Mg x1 y1 z1

(all other atoms in the Atoms block omitted here)

End

GeoVar

x1=.00000 F

y1=.00000 F

z1=.00000 F

x2=.00000 F

y2=1.72129 F

z2=1.82068 F

x3=.00000 F

y3=.00000 F

z3=-3.64100 F

x4=.00000 F

y4=-1.72130 F

z4=-1.82068 F

x5=.00000 F

y5=1.72130 F

z5=-1.82032 F

x6=.00000 F

y6=1.72130 F

z6=-5.46132 F

x7=2.53903

y7=.03004

z7=-3.50645

x8=2.50628

y8=-.07048

z8=-.10022

x9=2.63009

y9=3.50093

z9=-3.02634

...

Many of the coordinates have a 'F' after their initial value specification under Geovar, indicating that these

coordinates will be kept frozen during optimization.

The remaining initial value specifications are omitted here.

```
END
QMMM
  OPTIMIZE
    MAX_STEPS 3000
    MAX_GRADIENT 0.01
    METHOD BFGS
    PRINT_CYCLES 100
  SUBEND

  FORCE_FIELD_FILE champ_de_force.ff
```

The local file 'champ\_de\_force.ff' is used as force field file. Of course, this is the file we've just set up in the run script.

```
  OUTPUT_LEVEL=1
  WARNING_LEVEL=1
  ELSTAT_COUPLING_MODEL=1
```

```
MM_CONNECTION_TABLE
  1 Mg QM  2  4  5  8 58 60
```

...

Contents of the MM\_Connection\_Table block is omitted.

```
  SUBEND
  CHARGES
    1 .957
    2 -.608
    3 1.017
    4 -.411
    5 -.561
```

...

Initial charges are specified for (all) the atoms. Whether or not the charges on the QM (and LI) atoms are used depends on the type of electrostatic coupling between the QM and MM system. See the QMMM manual for details.

```
  SUBEND
END
```

```
Fragments
Ti t21.Ti
Cl t21.Cl
Mg t21.Mg
C t21.C
H t21.H
End
```

```
End Input
eor
```

# POST-ADF Property and utility programs

## NMR chemical shifts and spin-spin coupling constants

### HBr: NMR Chemical Shifts

*Sample directories:* adf/e\_HBr/ and adf/e\_HBr\_SO/

Computation of the NMR chemical shifts for HBr. The second sample uses spin-orbit relativistic corrections.

```
$ADFBIN/adf << eor
TITLE HBr non-relativistic
```

```
ATOMS
 1. H .0000 .0000 .0000
 2. Br .0000 .0000 1.4140
End
```

```
Basis
Type DZ
Core Large
End
```

```
XC
GGA Becke Perdew
End
```

```
End input
eor
```

The TAPE21 result file of ADF must be present under that name for the NMR calculation

```
mv t21.nmr TAPE21
```

The NMR program uses only one input (block) key NMR, currently. The subkeys specify what output is produced (OUT) and for which Nuclei the NMR data are computed and printed (NUC). See the User's Guide and the Utilities document for more details.

```
$ADFBIN/nmr << eor
NMR
  Out TENS
  Nuc 1 2
End
eor
```

The second run is like the first, except that it uses relativistic corrections, including Spin-Orbit terms. This implies that NOSYM symmetry *must* be used in the ADF calculation: the NMR program cannot handle symmetry calculations in combination with spin-orbit terms and will stop with an error message if you try to do so.

```
$ADFBIN/adf << eor
TITLE HBr relativistic spinorbit Pauli
```

## Atoms

```
1. H .0000 .0000 .0000
2. Br .0000 .0000 1.4140
```

End

## Basis

```
Type DZ
Core Large
End
```

Symmetry NoSYM

## XC

```
GGA Becke Perdew
End
```

Relativistic SpinOrbit Pauli

```
End input
eor
```

rm t12.rel

\$ADFBIN/nmr &lt;&lt; eor

```
NMR
OUT TENS
NUC 1 2
```

```
End
eor
```

## HgMeBr: NMR Chemical Shifts

*Sample directories:* `adf/e_HgMeBr_pnr/` (non-relativistic), `adf/e_HgMeBr_psc/` (Pauli scalar relativistic), `adf/e_HgMeBr_zso/` (ZORA relativistic *and* Spin-Orbit terms included)

NMR data are computed for the 1<sup>st</sup> and 3<sup>rd</sup> nucleus only. The UIK subkey is used to indicate that certain terms are to be included in the 'U-matrix', which goes into the first-order change of the MO's due to the applied magnetic field. See the documentation (Utilities) for more information.

The 'BEST' specification means that the mass-velocity and Darwin terms are included for a scalar relativistic calculation. In a non-relativistic run it has no meaning. In a spin-orbit run it would include the Fermi-contact term for the Pauli formalism, and the ZORA Spin-Orbit terms for a ZORA calculation.

```
$ADFBIN/nmr << eor
NMR
  U1K BEST
  NUC 1 3
END
eor
```

The other two calculations are similar, apart from the specification of the applicable relativistic features.

## CH<sub>4</sub>: NMR Chemical Shifts, SAOP potential

*Sample directories:* adf/e\_CH4\_SAOP/

Computation of the NMR chemical shifts for CH<sub>4</sub>, with the model potential SAOP.

Important: use SAVE TAPE10. This is necessary for SAOP, since the nmr program does not know about SAOP or other model potentials. On TAPE10 the SCF potential is written, which is read in by the nmr program.

Note: For SAOP one needs an all-electron basis set

```
$ADFBIN/adf << eor
xc
  model saop
end

Define
RCH = 1.085
XCH = sqrt(3)*(RCH/3)
End

Atoms
C 0 0 0
H XCH XCH XCH
H XCH -XCH -XCH
H -XCH XCH -XCH
H -XCH -XCH XCH
End

Basis
Type TZ2P
Core None
End

save TAPE10
End Input
eor

$ADFBIN/nmr << eor
NMR
  Out TENS
  Nuc 1 2
End
eor
```

## CO: NMR Chemical Shifts, SIC-VWN potential

*Sample directories:* adf/e\_CO\_fc\_SICVWN/

Computation of the NMR chemical shifts for CO, with the SIC-VWN potential.

Important: use SAVE TAPE10. This is necessary for SAOP, since the epr or nmr program does not know about SIC-VWN or other model potentials. On TAPE10 the SCF potential is written, which is read in by the epr or nmr program.

Note: adf with the SIC-VWN only runs serial correctly, and symmetry NOSYM is required.

Note: Both epr and nmr change TAPE10, TAPE21. Therefore use original TAPES from adf.

```
$ADFBIN/adf -n1 << eor
TITLE CO, SIC-VWN, Basis set TZ2P
Basis
  Type TZ2P
  Core Small
End
RELATIVISTIC Scalar Pauli
SYMMETRY NOSYM
ATOMS Z-mat
  1 C 0 0 0
  2 O 1 0 0 RCO
END
GEOVAR
  RCO 1.139719
END
INTEGRATION 6.0
XC
  LDA VWN
END
SICOEP
  IPRINT 1
  SELF 35
END
SAVE TAPE10
END INPUT
eor
```

```
cp TAPE21 t21
cp TAPE10 t10
```

```
$ADFBIN/epr -n1 << eor
NMRSHIELDING
  NUCLEI ALL
  OUTPUT
  SIZE LARGE
  SUBEND
END
SICOEP
  IPRINT 1
```

END

eor

rm TAPE10 TAPE15 TAPE21

cp t21 TAPE21

cp t10 TAPE10

\$ADFBIN/nmr << eor

NMR

U1K BEST

END

eor

## PF<sub>3</sub>: NMR Properties, Nucleus-independent chemical shifts

*Sample directory:* adf/e\_Nmr\_PF3/

The EPR/NMR program enables the calculation of so-called nucleus-independent chemical shifts. More details are available in the Properties Programs User's Guide.

In the ADF run, the Efield key is used to define points charges with zero charge. The GHOSTS key in the epr program then basically copies this block. For the interpretation of the results we refer to the literature.

```
...
Efield
  3.0 4.0 5.0 0.0
  1.0 2.0 3.0 0.0
End
...
eor

$ADFBIN/epr -n1 << eor
CLGEPR
  NUCLEI ALL
  GHOSTS
    3.0 4.0 5.0
    1.0 2.0 3.0
  SUBEND
  OUTPUT
    SIZE LARGE
  SUBEND
END

END INPUT
eor
```

## PF<sub>3</sub>: Comparison of NMR with EPR/NMR

*Sample directory: adf/e\_PF3\_nmr/*

This example uses both the NMR program and the EPR/NMR program, which is somewhat different from the NMR program used in the examples above. Please check the Property Programs User's Guide for a discussion on the advantages and disadvantages of the two implementations. This example explicitly compares the two.

```
$ADFBIN/nmr << eor
NMR
  U1K BEST
END
eor
```

The NMR program can currently formally be run in parallel. Due to certain single-CPU bottlenecks, this is hardly noticeable at the moment though. For this reason, there is currently limited advantage in using more than one CPU for the nmr program. Most other property programs can currently not be run in parallel at all and require the -n1 flag.

The output of the two NMR calculations should be virtually identical.

```
$ADFBIN/epr -n1 << eor
CLGEPR
  NUCLEI ALL
  OUTPUT
  SIZE LARGE
  SUBEND
END

END INPUT
eor
```

## VOCI3: NMR Properties

*Sample directory:* adf/e\_Nmr\_VOCI3/

After a scalar relativistic Pauli calculation in ADF, using NOSYM, the EPR/NMR program is invoked. The EPR/NMR program does not support ZORA at the moment.

```
$ADFBIN/adf << eor  
TITLE VOCI3
```

```
CHARGE 0
```

```
ATOMS Z-mat
```

```
 1 V 0 0 0  
 2 O 1 0 0 RVO  
 3 Cl 1 2 0 RVCi AOVCI  
 4 Cl 1 2 3 RVCi AOVCI 120.  
 5 Cl 1 2 3 RVCi AOVCI -120.  
END
```

```
INTEGRATION 5.0 5.0
```

```
GEOVAR
```

```
  RVO  1.584452  
  RVCi  2.164767  
  AOVCI 108.614124  
END
```

```
SYMMETRY NOSYM
```

```
RELATIVISTIC Scalar Pauli
```

```
XC
```

```
  LDA VWN  
END
```

```
FRAGMENTS
```

```
  V t21.V  
  O t21.O  
  Cl t21.Cl  
END
```

```
COREPOTENTIALS t12.rel &
```

```
  V 1  
  Cl 2  
  O 3  
END
```

```
END INPUT
```

```
eor
```

The NUCLEI key now specifies that all atoms are to be treated for NMR. Much output is demanded.

```
$ADFBIN/epr -n1 << eor  
CLGEPR  
  NUCLEI ALL  
  OUTPUT  
    SIZE LARGE  
  SUBEND  
END  
END INPUT  
eor
```

## C2H2: Nuclear Spin-spin coupling constants

*Sample directory adf/e\_CPL\_C2H2*

### Nonrelativistic calculation

A calculation of NMR nuclear spin-spin coupling constants (NSCCs).

As explained in the 'ADF Properties Programs' documentation, the quality of a calculation for spin-spin coupling constants, using the program 'CPL', depends largely on the preceding ADF calculation, which produces the Kohn-Sham orbitals and orbital energies, used as a starting point.

One of the quality-determining factors is the chosen basis set. It should be sufficiently flexible near the nucleus. Although the all-electron basis V is chosen in this example, it is recommendable to add more functions to the basis and fit sets near the nucleus in case of heavy elements. One could start from a ZORA/QZ basis for example.

The integration accuracy in the ADF calculation is chosen such that the region near the nuclei is described relatively more accurately than the rest of the molecule.

#### INTEGRATION

```
accint 5
accsph 6
end
```

The NOSYM symmetry currently needs to be specified in ADF to enable the CPL program to work correctly.

The first call to cpl is as follows:

```
$ADFBIN/cpl <<eor
maxmemoryusage 40
nmrcoupling
dso
pso
sd
scf convergence 1e-7
nuclei 1 2 3 4
nuclei 3 4
end
endinput
eor
```

The CPL program can run in parallel.

The maxmemoryusage 40 line should normally be superfluous, the default memory is sufficient for most cases. The keyword may be needed for calculations on large molecules though.

The specification of what needs to be calculated is given in the nmrcoupling block key.

In this first example, the SD subkey is left out, as this would lead to a very strong increase in the required CPU time. The SD subkey is included in the second CPL run. That subkey controls the calculation of the so-called spin-dipole term.

The subkeys `dso` and `pso` specify that, respectively, the diamagnetic and paramagnetic orbital terms will be calculated. The often dominant Fermi contact term (FC) is calculated by default and therefore does not have to be specified explicitly.

The `scf` convergence subkey, in this context, refers to the convergence for the solution of the coupled-perturbed Kohn-Sham equations which need to be solved to obtain spin-spin couplings.

The lines

```
nuclei 1 2 3 4
nuclei 3 4
```

that one coupled-perturbed Kohn-Sham calculation is performed where nucleus number 1 (according to the ordering in the ADF output) is the perturbing nucleus, and nuclei 2, 3, and 4 are the perturbed nuclei, and another coupled-perturbed Kohn-Sham calculation is performed where nucleus 3 is the perturbing nucleus and nucleus 4 is the perturbed nucleus.

The second CPL run also includes the spin-dipole (SD) term, through the `SD` subkey.

The output of the CPL program first contains a lot of general information, a summary of the specified input, and then produces the desired numbers:

It prints separately the different contributions (FC, DSO, PSO, SD) if specified in input and sums them up to a total number. Experimental NSCCs between two nuclei A and B are usually reported as  $J(A,B)$  in Hertz. From a computational point of view, the so-called reduced NSCCs  $K(A,B)$  are more convenient for comparisons. CPL outputs both. In this example, the Fermi-contact term is indeed dominant.

The first part of the output refers to the line

```
nuclei 1 2 3 4
```

then the same thing is done for the second similar line where nucleus 3 is the perturbing nucleus.

The output for the second CPL run looks very similar, but now the SD term is added to the Fermi contact term, resulting in much longer execution times.

## Scalar relativistic and spin-orbit calculations

The CPL program also enables calculations using scalar relativistic effects (ZORA) and/or spin-orbit effects.

Schematically, this requires the following changes to the input file with respect to a regular spin-orbit calculation and a nonrelativistic CPL calculation:

- steep (1s) functions may need to be added to the standard basis sets.
- the full-potential option for ZORA is needed in the create runs and all further runs:  
relativistic zora scalar full
- the molecular ADF calculation should contain the line  
relativistic zora full spinorbit
- the CPL input is unmodified with respect to the example given here. Please check the 'ADF Property Programs' document for details on relativistic input options.

## ESR / EPR properties

### TiF<sub>3</sub>: ESR Properties

*Sample directory:* adf/e\_ESR\_TiF3/

You calculate Electron Spin Resonance properties with the keywords ESR and QTENS. ESR is a block-type key and is used to compute the G-tensor or the Nuclear Magnetic Dipole Hyperfine interaction. QTENS is a simple key and invokes the computation of the Nuclear Electric Quadrupole Hyperfine interaction.

Proper usage of the key ESR requires that you do one of the following:

- (a) A Spin-Orbit calculation, spin-restricted, with exactly one unpaired electron, or
- (b) A Spin-Orbit calculation, spin-unrestricted in the collinear approximation, or
- (c) No Spin-Orbit terms and spin-*un*restricted.

In case (a) and (b) you obtain the G-tensor. In case (b) and (c) you get the Magnetic Dipole Hyperfine interaction.

Note: in case (a) the program also prints a Magnetic Dipole Hyperfine interaction data, but these have then been computed without the terms from the spin-density at the nucleus.

Note: in case (b) and (c) one can have more than one unpaired electron.

Note: in case (b) one has to use symmetry NOSYM.

Five calculations are performed:

- Scalar relativistic spin-unrestricted
- Spin-Orbit relativistic spin-restricted
- Scalar relativistic spin-restricted
- Scalar relativistic open shell spin-restricted
- Spin-Orbit relativistic spin-unrestricted collinear

After the preliminary calculations (DIRAC, to get the relativistic TAPE12 file with relativistic potentials, and the Create runs), we first calculate the Dipole Hyperfine interaction: a spin-unrestricted calculation without Spin-Orbit coupling.

```
$ADFBIN/adf << eor
title TiF3 relativistic open shell unrestricted
noprnt sfo,frag,functions
```

```
DEFINE
RTIF = 1.780
RY = RTIF*SQRT(3)/2
END
```

```
esr
end
```

```
qtens
```

```
atoms
Ti 0 0 0
F RTIF 0 0
```

```

F -RTIF/2 RY 0
F -RTIF/2 -RY 0
end

fragments
Ti t21.ti
F t21.f
end

xc
GGA Becke Perdew
end

charge 0 1
unrestricted

relativistic scalar zora
Corepotentials t12.rel &
Ti 1
F 2
end

end input
eor

```

Then, for the same molecule, we compute the G-tensor in a Spin-Orbit run (spin-restricted).

The here-computed and printed Dipole Hyperfine interaction misses the terms from the spin-density at the nucleus: compare with the outcomes from the first calculation.

In each of the calculations, the QTENS key invokes the computation of the Electric Quadrupole Hyperfine interaction.

Note that an *all-electron* calculation is carried out. This is relevant for the computation of the A-tensor, the nuclear magnetic dipole hyperfine interaction, where an accurate value of the spin-polarization density at the nucleus is important. For the G-tensor (and also for the Q-tensor) this plays a minor role, but for reasons of consistency both calculations use the same basis set and (absence of) frozen core.

```

$ADFBIN/adf << eor
title Tif3 relativistic spinorbit open shell restricted
noprnt sfo,frag,functions

DEFINE
RTIF = 1.780
RY = RTIF*SQRT(3)/2
END

esr
end

qtens

atoms
Ti 0 0 0
F RTIF 0 0
F -RTIF/2 RY 0

```

```
F -RTIF/2 -RY 0
end
```

```
fragments
```

```
  Ti t21.ti
  F  t21.f
end
```

```
xc
```

```
  GGA Becke Perdew
end
```

```
relativistic spinorbit zora
```

```
Corepotentials t12.rel &
  Ti 1
  F 2
end
```

```
end input
```

```
eor
```

Next a scalar relativistic spin-restricted calculation is performed. The TAPE21 of this calculation is saved as a fragment in the next spin-unrestricted calculation, using only 1 SCF iteration, which is a way to get the scalar relativistic spin-restricted open shell result for the magnetic dipole hyperfine interaction.

```
$ADFBIN/adf << eor
```

```
title TiF3 scalar relativistic restricted
noprnt sfo,frag,functions
```

```
DEFINE
```

```
  RTIF = 1.780
  RY = RTIF*SQRT(3)/2
END
```

```
atoms
```

```
  Ti 0 0 0
  F  RTIF 0 0
  F -RTIF/2 RY 0
  F -RTIF/2 -RY 0
end
```

```
fragments
```

```
  Ti t21.ti
  F  t21.f
end
```

```
xc
```

```
  GGA Becke Perdew
end
```

```
relativistic scalar zora
```

```
Corepotentials t12.rel &
  Ti 1
  F 2
end
```

```
end input
eor

mv TAPE21 t21.TiF3
rm logfile

$ADFBIN/adf << eor
title TiF3 scalar relativistic open shell restricted
noprnt sfo,frag,functions

DEFINE
RTIF = 1.780
RY = RTIF*SQRT(3)/2
END

esr
end

qtens

atoms
Ti 0 0 0 f=TiF3
F RTIF 0 0 f=TiF3
F -RTIF/2 RY 0 f=TiF3
F -RTIF/2 -RY 0 f=TiF3
end

fragments
TiF3 t21.TiF3
end

xc
GGA Becke Perdew
end

charge 0 1
unrestricted

scf
iter 0
end

relativistic scalar zora
Corepotentials t12.rel &
Ti 1
F 2
end

end input
eor
```

Finally a spin-orbit coupled spin-unrestricted calculation is performed using the collinear approximation. Note that symmetry NOSYM is used.

```
$ADFBIN/adf << eor
title TiF3 relativistic spinorbit open shell unrestricted collinear
noprnt sfo,frag,functions

DEFINE
  RTIF = 1.780
  RY = RTIF*SQRT(3)/2
END

esr
end

qtens

symmetry nosym
unrestricted
collinear

atoms
  Ti 0 0 0
  F RTIF 0 0
  F -RTIF/2 RY 0
  F -RTIF/2 -RY 0
end

fragments
  Ti t21.ti
  F t21.f
end

xc
  GGA Becke Perdew
end

relativistic spinorbit zora
Corepotentials t12.rel &
  Ti 1
  F 2
end

end input
eor
```

## VO: collinear approximation

*Sample directory:* `adf/e_VO_collinear/`

The ESR parameters of VO are calculated with the collinear approximation for unrestricted Spin-Orbit coupled calculations. In this example the VO-molecule has three unpaired electrons.

You calculate Electron Spin Resonance properties with the keywords ESR and QTENS. ESR is a block-type key and is used to compute the G-tensor or the Nuclear Magnetic Dipole Hyperfine interaction. QTENS is a simple key and invokes the computation of the Nuclear Electric Quadrupole Hyperfine interaction.

Proper usage of the key ESR requires that you do one of the following:

- (a) A Spin-Orbit calculation, spin-restricted, with exactly one unpaired electron, or
- (b) A Spin-Orbit calculation, spin-unrestricted in the collinear approximation, or
- (c) No Spin-Orbit terms and spin-*un*restricted.

In case (a) and (b) you obtain the G-tensor. In case (b) and (c) you get the Magnetic Dipole Hyperfine interaction.

Note: in case (a) the program also prints a Magnetic Dipole Hyperfine interaction data, but these have then been computed without the terms from the spin-density at the nucleus.

Note: in case (b) and (c) one can have more than one unpaired electron.

Note: in case (b) one has to use symmetry NOSYM.

Two calculations are performed:

- Scalar relativistic spin-unrestricted
- Spin-Orbit relativistic spin-unrestricted collinear

After the preliminary calculations (DIRAC, to get the relativistic TAPE12 file with relativistic potentials, and the Create runs), we first calculate the Dipole Hyperfine interaction: a spin-unrestricted calculation without Spin-Orbit coupling.

Note that one has to use ALLPOINTS in the calculation for a linear molecule to get results for the nuclear magnetic dipole hyperfine interaction.

```
$ADFBIN/adf << eor
```

```
Atoms
```

```
V 0 0 0
```

```
O 0 0 1.589
```

```
End
```

```
XC
```

```
GGA Becke Perdew
```

```
End
```

```
esr
```

```
end
```

```
qtens
```

```
allpoints
```

```
unrestricted
```

```
charge 0 3
```

```
Relativistic Scalar ZORA
CorePotentials t12.rel &
V 1
O 2
End
```

```
integration 5
```

```
Fragments
V t21.V
O t21.O
End
End input
eor
```

Then a spin-orbit coupled spin-unrestricted calculation is performed using the collinear approximation. Note that symmetry NOSYM is used.

```
$ADFBIN/adf << eor
Atoms
V 0 0 0
O 0 0 1.589
End
```

```
XC
GGA Becke Perdew
End
```

```
esr
end
qtens
```

```
symmetry nosym
unrestricted
collinear
```

```
Relativistic Spinorbit ZORA
CorePotentials t12.rel &
V 1
O 2
End
```

```
integration 5
```

```
Fragments
V t21.V
O t21.O
End
End input
eor
```

## Ge<sup>+</sup> and H<sub>2</sub><sup>+</sup>: EPR spectrum

*Sample directories:* `adf/e_Epr_Ge2+` and `adf/e_Epr_H2+`

The NMR/EPR program gives functionality that partially overlaps and partially differs from the ESR keyword inside ADF. Please check the ADF and Property Programs User's Guides for details.

In this example, a scalar relativistic Pauli calculation is performed (ZORA is not implemented in this program).

The preparatory ADF calculation is run serially, without the use of symmetry, and a high numerical integration accuracy. The newly implemented revised PBE functional is invoked. The implementation allows spin-unrestricted high-spin inputs and goes beyond the ESR implementation within ADF itself, in this sense.

The H<sub>2</sub><sup>+</sup> example is very similar and in fact a bit simpler, so it will not be discussed separately here.

```
$ADFBIN/adf << eor
```

```
TITLE Ge2+, scf
```

```
SYMMETRY NOSYM
```

```
UNRESTRICTED
```

```
CHARGE +1 3
```

```
ATOMS Cart
```

```
 1 Ge 0.0000 0.0000 -1.2344  
 2 Ge 0.0000 0.0000  1.2344
```

```
END
```

```
INTEGRATION 6.0
```

```
FRAGMENTS
```

```
  Ge t21.Ge
```

```
END
```

```
COREPOTENTIALS t12.rel &
```

```
  Ge 1
```

```
END
```

```
XC
```

```
  GGA revPBE
```

```
END
```

```
RELATIVISTIC Scalar Pauli
```

```
END INPUT
```

```
eor
```

The EPR calculation itself then has a fairly simple input. It uses the TAPE21 file generated by ADF. The full EPR G-tensor is printed, including an extensive analysis for the contribution of different terms.

The input line

```
NUCLEI NONE
```

implies that no NMR calculation is requested.

```
$ADFBIN/epr -n1 << eor
```

```
CLGEPR
```

```
EPRGT
```

```
SUBEND
```

```
NUCLEI NONE
```

```
OUTPUT
```

```
  EPRSIZE LARGE
```

```
SUBEND
```

```
END
```

```
END INPUT
```

```
eor
```

## Analytic second derivatives

### CN: Analytic second derivatives

*Sample directory:* adf/e\_CN\_SecDeriv/

The ADF2002.01 version features analytic second derivatives (SD) for the first time. This initial implementation still has severe limitations though, both in terms of speed, as in terms of user-friendliness of the output and the number of available options. For most ADF users, it will still be recommendable to use the finite difference implementation which can be invoked through the FREQUENCIES subkey in the GEOMETRY block (see ADF User's Guide for details).

As in other property programs, first a preparatory ADF calculation needs to be done. This calculation will usually take much less time than the SD calculation. There are a number of restrictions to the ADF run. The line

```
symmetry NOSYM
```

is mandatory. The X-alpha potential is specified. Although the programs will accept other potentials, such as GGAs, the user should realize that only the X-alpha functional is currently implemented in the SD program. Although the practical results resulting from a mixed GGA/X-alpha calculation may be very good, please realize that the resulting frequencies will neither be identical to pure GGA, nor the X-alpha frequencies.

Further note that in this example, and many others, the new convenient BASIS keyword is used, which circumvents the need to specify separate Create runs for the atoms.

A high accuracy is specified for the numerical integration to be sure of reliable results. In general, it seems advisable to use high accuracy for heavy nuclei at the moment, whereas default integration accuracy is usually sufficient for light atoms. Further, high integration accuracy is more needed in the atomic spheres than in the rest of the molecule. A cost-effective solution may therefore be to specify a higher integration accuracy in the spheres only (using the accsph subkey of the INTEGRATION keyword).

```
$ADFBIN/adf << eor  
title CN
```

```
atoms  
N -1.3 0.0 0.0  
C 0.0 0.0 0.0  
end
```

```
Basis  
Type DZ  
Core None  
End
```

```
charge -1
```

```
XC  
LDA Xonly  
End
```

```
integration 6.0
```

symmetry NOSYM

End input  
eor

After the ADF calculation is finished, the SD program reads the TAPE21 result file (as well as possibly other ADF result files) and starts its lengthy calculation. The print keyword obviously gives timing information. The SD keyword specifies that both the GRADIENT (analytic first derivative of energy with respect to nuclear displacements), the HESSIAN (analytic second derivative), and the DIPOLE derivatives are to be calculated and printed. The latter give access to the IR intensities, whereas the HESSIAN gives access to the IR frequencies.

The SD program can run in parallel.

```
$ADFBIN/sd << eor  
print timing  
integration 6.0  
SD  
  CALC HESSIAN DIPOLE GRADIENT  
END  
eor
```

## CH4: Analytic second derivatives

*Sample directory:* adf/e\_CH4\_SecDeriv/

The CH<sub>4</sub> example for the analytic second derivatives is very similar to the CN example. The comments are therefore restricted to a few statements.

```
$ADFBIN/adf << eor  
title CH4
```

```
Define  
ZERO = 0.0  
RCH = 1.0850  
X1 = ZERO  
Y1 = ZERO  
Z1 = ZERO  
X2 = sqrt(3)*(RCH/3)  
Y2 = -sqrt(3)*(RCH/3)  
Z2 = sqrt(3)*(RCH/3)  
X3 = sqrt(3)*(RCH/3)  
Y3 = sqrt(3)*(RCH/3)  
Z3 = -sqrt(3)*(RCH/3)  
X4 = -sqrt(3)*(RCH/3)  
Y4 = sqrt(3)*(RCH/3)  
Z4 = sqrt(3)*(RCH/3)  
X5 = -sqrt(3)*(RCH/3)  
Y5 = -sqrt(3)*(RCH/3)  
Z5 = -sqrt(3)*(RCH/3)  
End
```

```
Atoms  
1. C X1 Y1 Z1  
2. H X2 Y2 Z2  
3. H X3 Y3 Z3  
4. H X4 Y4 Z4  
5. H X5 Y5 Z5  
End
```

```
Basis  
Type TZP  
Core None  
End
```

```
XC  
LDA Xonly  
End
```

```
integration 4.0
```

```
symmetry NOSYM
```

```
End input
```

eor

This time more output is given in the SD part. The input now specifies that 1 iteration is to be done in the coupled-perturbed Kohn-Sham equations before the Hessian is printed. More iterations given a better converged result for the Hessian, but at the cost of increased computer time, which is roughly proportional to the number of iterations.

```
$ADFBIN/sd << eor
print timing
SD
CALC GRADIENT HESSIAN DIPOLE
MAX_CPKS_ITERATIONS 8
CHECK_CPKS_FROM_ITERATION 1
U1_ACCURACY 0.0001
END
eor
```

## HI: ZORA Analytic second derivatives

*Sample directory:* adf/e\_HI\_SecDer\_ZORA/

The main difference of this example to the previous examples is that a ZORA Hessian is calculated in this example, through the line (in the ADF calculation only!):

RELATIVISTIC scalar ZORA

Furthermore, the suggestion to use high integration accuracy in the atomic spheres only is shown explicitly here.

Saving the ADF result file TAPE10 results in a slightly reduced amount of recomputation in SD.

```
$ADFBIN/adf << eor
TITLE HI scalar, ZORA,
DEFINE
ZERO = 0.0
R = 1.6090
X1 = ZERO
Y1 = ZERO
Z1 = R
X2 = ZERO
Y2 = ZERO
Z2 = ZERO
END
ATOMS
I X2 Y2 Z2
H X1 Y1 Z1
END
XC
LDA Xonly
END
RELATIVISTIC scalar ZORA
COREPOTENTIALS t12.rel &
H 1
I 2
END
symmetry NOSYM
FRAGMENTS
H t21H
I t21I
END
integration
accint 4.0
accsph 6.0
end
end input
eor
```

In the SD program, more output on details of the calculation is printed.

```
$ADFBIN/sd << eor
```

```
print computation
print numintpar
print timing
integration
  accint 4.0
  accsph 6.0
end
SD
  CALC HESSIAN GRADIENT DIPOLE
  MAX_CPKS_ITERATIONS 8
  CHECK_CPKS_FROM_ITERATION 1
  U1_ACCURACY 0.0001
END
eor
```

## Post-ADF analysis utilities

### NO<sub>2</sub>: Contour Plots using *Densf* and *Cntrs*

Sample directory: *adf/e\_Cntrs\_NO2/*

This example illustrates using the utility programs *cntrs* and *densf*. See the Utilities document for details.

```
$ADFBIN/adf << eor  
title NO2
```

```
atoms
```

```
N  0      0      0  
O  1.009356  0      0.464189  
O -1.009356  0      0.464189  
end
```

```
Basis
```

```
Type DZ  
Core Small  
End
```

```
unrestricted  
charge 0 1
```

```
endinput  
eor
```

After the normal ADF calculation on NO<sub>2</sub> has been completed, the utility program *densf* is executed to generate a TAPE41 file with user-specified items evaluated in a regular, user-specified grid.

The TAPE21 on which *densf* operates must be present as a local file with name TAPE21.

```
$ADFBIN/densf << eor  
density  scf ortho frag  
fitdensity  scf ortho frag  
orbitals  scf  
alpha  
a1  1 2  
a2  1  
b1  2  
beta  
a2  1  
b1  1 2  
b2  1  
end  
coulpot  frag ortho scf  
grid  
-7.5 -7.5 0.0  
51 51  
1.0 0.0 0.0 15.00
```

```
0.0 1.0 0.0 15.00
end
end input
eor
```

The charge density values in the grid are requested for all available types of density: exact and fitted, for the initial (sum-of-fragments), intermediate (orthogonalized fragments, see the ADF User's Guide) and final (SCF) situation.

Several SCF molecular orbitals are computed by specifying their indices in the energy-ordered list (a separate list for each symmetry subspecies).

The coulomb potentials (again: for sum-of-fragments, orthogonalized fragments, and SCF) are generated.

The grid is defined by an 'origin', the numbers of points in all independent grid directions and the direction vectors with the total grid size in each direction separately.

Since there are only two 'numbers-of-points' (51 each) a 2-dimensional grid is generated. 1D and 3D grids are also possible. See the Utilities document for a more detailed survey of the available options.

The result of the *densf* run is a file *tape41* (binary, KF). This contains all computed data. *tape41* can be used by *cntrs* to generate plot data.

```
$ADFBIN/cntrs << eor
scan
0.02 0.05 0.10 0.2 0.5 0.0 -0.02 -0.05 -0.10 -0.2 -0.5
end
dash 0.2
file cont.d
SCF%Density_A
SCF%Density_B
endinput
eor
```

In this example eleven (11) scan values are defined to draw contours for, with a dash length of 0.2 bohr.

An ascii file *cont.d* will be opened by *cntrs* on which the specified items (SCF-densities for spin-up and spin-down) will be combined (by default: simply added) into one quantity.

For this quantity the contour lines that correspond to the specified scan values are stored. See the Utilities document for precise specifications and options.

```
gnuplot << eor

set term dumb 100 80
set output "outplot"
plot "cont.d" using 1:2 with lines
eor
```

```
cat outplot
```

The public domain software *gnuplot* (not included in the *adf* package) is applied here to display the result from *cntrs*. The resulting picture on your screen (if you have *gnuplot* available) looks like

```
10 *+
*
*          *****
*          *****          *****
```

```
*          **          *****          **
*          ** ***** ***** ***** **
*          ** ** ***** ** ** ***** ** *
*          * *****      ** **      * *
*          * * * *****      ***** ** *
8 *+      * * * *      * * * *      * * * *
*          * * * *      **      **      * * *
*          * * * *      * * * *      * * *
*          * * * * ** * * * *      *** ** *
*          * * * *      * * * *      * * *
*          ** ** **      *      **      * * *
*          * * * *      *****      * * *
*          * * * * ***** ***** **
*          ** ** ***** ***** ** *
6 *+      ** ***** ***** ***** **
*          ***          *****          ***
*          *****          *****
*          *****
```

## C<sub>2</sub>H<sub>2</sub>: Localization of Molecular Orbitals

Sample directory: `adf/e_Cntrs.LocOrb_C2H2/`

An *illustration* of the computation of localized molecular orbitals in C<sub>2</sub>H<sub>2</sub>.

The delocalized molecular orbitals as they result from the scf are localized in two different ways. In the first the three  $\sigma$  bonds are recombined only among themselves (no  $\pi$  bonds are mixed in), yielding two equivalent localized CH  $\sigma$  bonds and one localized  $\sigma$  bond. In a second step the localization of the remaining bond (the two  $\pi$ 's) is performed, but this produces nothing new since no combination of the two  $\pi$ 's is more localized than they are already by themselves.

```
$ADFBIN/adf << eor
title C2H2, localization Sigma and Pi separately
```

```
Atoms
C 0 0 .63
C 0 0 -.63
H 0 0 1.63
H 0 0 -1.63
END
```

```
Basis
Type TZP
Core Small
End
```

```
LocOrb
alfa 4 5
alfa 1 2 3
END
```

```
integration 4.0
```

```
end input
eor
```

In the first localization cycle the  $\pi$ -orbitals are left out: #4 and #5 in the list of all occupied valence MOs: first 3 MOs of the first irreducible representation (s), then the 2 from the second irrep ( $\pi$ ). In the second localization step the first three (meanwhile localized) orbitals are kept aside.

With *densf* the local orbitals can be computed in a user-defined grid (for plotting purposes). *densf* requires a file with name TAPE21.

```
$ADFBIN/densf << eor
```

```
Grid
0. -5. -5.
100 100
0. 0. 1. 10.
0. 1. 0. 10.
End
```

```
Orbitals Local
  1 2 3 4 5
End
```

```
END INPUT
eor
```

The program *cntrs* is applied to process the *densf* result file TAPE41.

```
$ADFBIN/cntrs << eor
```

```
SCAN
  0.01 0.02 0.04 0.08 0.16 0.32 0.64 1.28
 -0.01 -0.02 -0.04 -0.08 -0.16 -0.32 -0.64 -1.28
END
```

```
file ctr.a1
LocOrb%1 1.00
```

```
file ctr.a2
LocOrb%2 1.00
```

```
file ctr.a3
LocOrb%3 1.00
```

```
file ctr.a4
LocOrb%4 1.00
```

```
file ctr.a5
LocOrb%5 1.00
```

```
END INPUT
eor
```

Again, *gnuplot* may be used to display the result on your screen.

```
$gnuplot << eor
set term dumb 100 80
set output "outplot"
plot "ctr.a1" using 1:2 with lines
plot "ctr.a2" using 1:2 with lines
plot "ctr.a3" using 1:2 with lines
plot "ctr.a4" using 1:2 with lines
plot "ctr.a5" using 1:2 with lines
eor
```

```
cat outplot
```

This results in 5 pictures, the first one looking like:

```
*****
****          ****
***           ***
```

```

          **  *****  *****  **
        **  ***          ***  **
       **  ***          ***  **
      **  **  *****          **
     *  **  ****          ****  **
    **  *  ***          ***  *  **
   *  **  **          **  **  *
  *  *  ***  *****          ***  *  *
 *  *  *  ***          ***  *  *  *
**  **  **          **  **  *  *
*****  *  *  *          *  *  *  *
**  **  **  *  *  *  *****  *  *  *
*  **  **  *  *  *  *  ****  ****  **  *  *  *
*  **  *  ***  *  *  *  ****  ****  *  *  ***
**  ****  *  *****  *  **          **  *  *****
*  ***  *****  **  *****  *          *  *****  *
**  **          *****  *****  *          *  *****  *****
*  *          *  *****  *****  *****  *****
**  *  **          *****  **          **  *****
*  *  **  ***  **  *****  *          *  *****  *****
**  **  **          *****  *  **          **  *****  *
**          **  ***  *  *  ***          ***  *  *  ***
*  **  **  *  *  *  *  *  ****  ****  **  *  *  *
***  **  *  *  *  *  *****          *  *  *  *
*****          *  *  *  *          *  *  *  *
          *  **  **          **  **  *  *
          *  *  *  ***          ***  *  *  *
          *  *  ***  *****          ***  *  *
          *  **  **          **  **  *
          **  *  ***          ***  *  **
          *  **  ****          ****  **  *
          **  **  *****          **  **
          **  ***          ***  **
          **  *****          ****  **
          **  *****          **
          ***          ***
          ****          ****
          *****

```

The second illustration of the computation of localized molecular orbitals in  $C_2H_2$  combines directly all MOs ( $\sigma$  and  $\pi$ ). This yields 3 equivalent 'banana' bonds, mixtures of  $\sigma$  and  $\pi$ , and two equivalent pure  $\sigma$  bonds.

```
$ADFBIN/adf << eor
```

```
title C2H2  localization without frozen orbitals
```

Atoms

C 0 0 .63

C 0 0 -.63

H 0 0 1.63

H 0 0 -1.63

end

fragments

C t21.C

H t21.H

end

integration 4.0

locorb

end

end input

eor

## Cu<sub>4</sub>CO: Density of States

Sample directory: `adf/e_DOS_Cu4CO/`

This sample illustrates the DOS property program to compute density-of-states data, for energy-dependent analysis.

First, the Cu<sub>4</sub>CO molecule is calculated (ADF), using single-atom fragments.

```
$ADFBIN/adf <<eor
title Cu4CO (3,1) from atoms

units
length bohr
end

define
rCu=2.784
end

atoms
1. Cu   rCu   0.0      0.0
2. Cu  -rCu/2  rCu*sqrt(3)/2  0.0
3. Cu  -rCu/2 -rCu*sqrt(3)/2  0.0
4. Cu   0.0   0.0      -rCu*sqrt(2)
5. C    0.0   0.0      2.65
6. O    0.0   0.0      4.91
end

fragments
Cu   t21.Cu
C    t21.C
O    t21.O
end

XC
  GGA PostSCF  Becke Perdew
END

endinput
eor
```

The PostSCF feature in the specification of the XC functional is used: the 'Becke-Perdew' GGA corrections are not included self-consistently but applied to the energy evaluation after the self-consistent LDA solution has been obtained.

```
mv TAPE21 t21.Cu4COh
```

The TAPE21 result file is saved for later usage.

```
cp t21.Cu4COh TAPE21
```

The utility program `dos` requires a TAPE21 file, with that name.

```
$ADFBIN/dos << eor
file dostxt

energyrange npoint=36 e-start=-25 e-end=10

tdos

gpdos
  a1 10 11 12 13 14
  e1:1 10 11 12 13 14
  e1:2 10 11 12 13 14
end

pdos
  a1 13 15
  a2 3 4
end

opdos
  bas 32
  SUBEND
  bas 1 2 6 7 32 33 34
end

end input
eor
```

Computed is the total density of states as well as various kinds of *partial* densities of states. You may feed the results into a plotting program like *gnuplot*. The result is not displayed here. See the *Utilities* document for more detailed info about *dos*.

# Input for third party Software

## **adf2aim: convert an ADF TAPE21 to WFN format (for Bader analysis)**

*Sample directory:* adf/e\_AIM\_HF/

ADF utility adf2aim (original name rdt21) developed by Xavi López, Engelber Sans and Carles Bo (see [http://www.quimica.urv.es/ADF\\_UTIL/](http://www.quimica.urv.es/ADF_UTIL/)):

**rdt21:** convert an ADF TAPE21 to WFN format (for Bader analysis)

This program rdt21 is now called adf2aim and is part of the ADF package, starting from ADF2004.01.

The WFN file is an input file for the third party program Xaim (see <http://www.quimica.urv.es/XAIM> for details), which is a graphical user interface to programs that can perform the Bader analysis.

Usage of adf2aim:

```
$ADFBIN/adf <<eor  
TITLE HF
```

```
ATOMS
```

```
1. H .0000 .0000 .0000  
2. F .0000 .0000 0.917  
End
```

```
Basis  
End
```

```
End input  
eor
```

```
$ADFBIN/adf2aim TAPE21  
echo 'Contents of rdt21.res:'  
cat rdt21.res  
echo 'Contents of WFN:'  
cat WFN
```

## NBO analysis: adfnbo, gennbo

*Sample directory:* adf/e\_H2O\_ADFNBO/

Dr. Autschbach, SCM, and Prof. Weinhold have collaborated to prepare a simple input file generator, called adfnbo, for the GENNBO program of Prof. Weinhold's Natural Bond Orbital (NBO) 5.0 package:

<http://www.chem.wisc.edu/~nbo5>

The GENNBO executable is included in the ADF distribution and can be enabled via the license file for all those who buy an NBO manual from either the NBO authors or from SCM ([info@scm.com](mailto:info@scm.com)).

Usage:

```
$ADFBIN/adf <<eor
```

```
Title simple NBO example for water
```

```
Atoms  Z-Matrix
O  0 0 0
H  1 0 0  0.9
H  1 2 0  0.9 100
End
```

```
Basis
CORE NONE
TYPE DZ
End
```

```
FULLFOCK
AOMAT2FILE
SAVE TAPE15
SYMMETRY NOSYM
```

```
End Input
eor
```

```
$ADFBIN/adfnbo <<eor
eor
```

```
$ADFBIN/gennbo < FILE47
```

A File named FILE47 is generated by adfnbo which is an input file for the general NBO program gennbo. ADF needs to write some data to file, which is done by including these keywords in the adf input file:

```
FULLFOCK
AOMAT2FILE
SAVE TAPE15
SYMMETRY NOSYM
```

## GENNBO

This section contains a brief summary of the capabilities of GENNBO, made available by Prof. Weinhold.

GENNBO implements most capabilities of the full NBO 5.0 program suite as described on the NBO website:

<http://www.chem.wisc.edu/~nbo5>

These include determination of natural atomic orbitals (NAOs), bond orbitals (NBOs), and localized MOs (NLMOs), as well as the associated NPA (atomic charges and orbital populations) and NRT (resonance structures, weightings, bond orders) valence descriptors, for a wide variety of uncorrelated and correlated (variational, perturbative, or density functional) theoretical levels. GENNBO-supported options include all keywords except those explicitly requiring interactive communication with the host electronic structure system (viz., \$DEL deletions, NEDA, NCS, NJC). The GENNBO program typically sits conveniently on the PC desktop, ready to analyze (or re-analyze at will, with altered options) the final results of a complex ADF calculation performed on a remote cluster.

GENNBO "communicates" with the original ADF calculation through an archive file (JOB.47 file, preserving all necessary details of the final density) that is initially generated by ADF and subsequently becomes the input file for GENNBO. The .47 file contains a standard \$NBO ... \$END keylist that can be edited with a standard word processor or text editor to include chosen NBO keyword options, just as though they might have appeared in the original input stream of an interactive ADFNBO run. The stand-alone GENNBO program therefore allows many alternative NBO analysis options to be explored at leisure, without costly re-calculation of the wavefunction.

# Sample Runs with BAND

## NaCl: Bulk Crystal

*Sample directory:* band/e\_NaCl/

A bulk crystal computation for Sodium Chloride (common salt), with a subsequent DOS analysis, using a Restart facility to use the results from a preceding calculation.

Calculations on periodic systems are carried out with the BAND program. Its input format has recently been changed substantially. It is now more similar in style to ADF. Old BAND input files are no longer compatible with the new version however.

The BAND input still follows slightly different conventions from the ADF input, for historical reasons.

The COMMENT keyword allows users to provide some information about the run which may be of use later. Usually a brief summary of the run is given here.

Numerical integration precision is controlled with the key Accuracy (in ADF: Integration)

The accuracy for integrals over the Brillouin Zone is set by the Kspace key. The latter should, generally, take as value an *odd* number (3, 5...) to invoke the accurate *quadratic* tetrahedron integration procedure. For *even* values it will revert to the *linear* tetrahedron method, which is almost always inferior in accuracy.

The key Angstroms specifies that geometric data, such as lattice constants are in angstrom units.

Since there are 3 data records in the Lattice block, the calculation will assume 3-dimensional periodicity, with lattice vectors as indicated. Note that lattice vectors are undefined up to linear combinations among themselves. Internally, the program will recombine the input vectors so as to minimize the size of the actually used vectors.

The input line Coordinates NATURAL means that atomic positions are input as coefficients in terms of the lattice vectors, rather than as absolute (Cartesian) coordinate values.

For each of the atoms in the calculations, Na and Cl here, there must be data blocks to specify various items. First, their positions in the crystal unit cell (key Atoms). Second, the single isolated atom computation that will serve as start-up (Dirac). Third, any Slater-type orbital basis functions (BasisFunctions) for that atom. Fourth, the fit functions (FitFunctions) for the calculation of the Coulomb potential and. The third item (BasisFunctions) is optional and not present in this example.

It is recommended to include the numerical atomic orbitals that are computed by the Herman-Skillman type subprogram DIRAC as basis functions for the periodic structure calculation. This is effectuated by putting the word VALENCE in the Dirac data blocks. If that is done, additional STO basis functions (key BasisFunctions) are optional and are used to increase the basis set flexibility. In absence of the numerical (DIRAC/VALENCE) orbitals, a minimal STO set is necessary of course, lest we wouldn't have any basis set at all.

In an equivalent ADF calculation, basis and fit functions would be provided through the Create runs, which pick up the basis and fit functions from a database file. The Create runs would also serve to provide the start-up density, as the DIRAC runs do in BAND.

The basis and fit sets that one has to insert into the BAND input files can be taken from the corresponding ADF database. Note, however, that ADF does not use any numerical orbitals. Since it is recommended to include such numerical orbitals in a BAND calculation, one has to adjust the STO-type basis set for BAND, in

comparison with ADF, so as to avoid linear dependency with the numerical orbitals. As a general guideline: for each of the included numerical orbitals (the occupied valence orbitals of the DIRAC calculation), one should remove one STO of the appropriate (n,l)-value. This keeps the overall size and flexibility of the basis at the same level and is usually sufficient to avoid dependency troubles.

The RUNKF key, early in the input, specifies that this standard result file from BAND must be saved under the name "t21.NaCl". This file will be used in the follow-up calculation of Density-of-States properties.

Note, finally, that the data blocks of block type keys in the input for BAND end with a record 'END', as in ADF, whereas previously '\*\*\*' was used in BAND to end a record.

```
$ADFBIN/band << eor
```

```
Title Title NaCl (from neutral atoms)
```

```
Comment
```

```
Technical
```

```
Hybrid K space integration (3D)
```

```
Low real space integration accuracy
```

```
Natural coordinates
```

```
Lengths in Angstrom
```

```
Parameters Dirac procedure
```

```
Features
```

```
Lattice : 3D
```

```
Unit cell : 2 atoms
```

```
Basis : NO w/ core
```

```
Options : Save restart file
```

```
End
```

```
MaxMemoryUsage 20
```

```
Save RUNKF ! for (DOS) restart purposes
```

```
Accuracy 3.5
```

```
Kspace 3
```

```
Units
```

```
Length Angstrom
```

```
End
```

```
Lattice
```

```
0 2.75 2.75
```

```
2.75 0 2.75
```

```
2.75 2.75 0
```

```
End
```

```
ATOMS NA
```

```
0
```

```
End
```

```
Coordinates Natural
```

```
Atoms Cl
```

```
.5 .5 .5
```

```
End
```

```
AtomType Na
```

```
Dirac Na
  4 1
  Radial 2000
  RMin 1E-4
  RMax 60
  VALENCE
  1 0
  2 0
  2 1
  3 0 1.0
SubEnd
```

```
FitFunctions
  1 0 18.9
  2 0 30.3
  2 0 15.5
  3 0 14.9
  3 0 8.9
  4 0 7.8
  4 0 5.1
  4 0 3.3
  5 0 2.8
  5 0 1.9
  5 0 1.3
  2 1 14.3
  3 1 9.9
  4 1 6.7
  4 1 3.4
  5 1 2.4
  5 1 1.3
  3 2 10.5
  4 2 5.4
  5 2 3.0
  5 2 1.3
  4 3 5.8
  5 3 1.7
  5 4 2.0
SubEnd
End
```

```
AtomType Cl
Dirac Cl
  5 3
  VALENCE
  1 0
  2 0
  2 1
  3 0
  3 1 5.0
SubEnd
```

```
FitFunctions
  1 0 29.1
  2 0 49.5
```

```

2 0 26.1
3 0 25.8
3 0 15.8
4 0 14.2
4 0 9.4
4 0 6.2
5 0 5.4
5 0 3.8
5 0 2.6

```

```

2 1 21.2
3 1 16.5
4 1 12.4
4 1 6.8
5 1 5.1
5 1 3.1

```

```

3 2 16.6
4 2 9.4
5 2 5.5
5 2 2.6

```

```

4 3 8.7
5 3 3.3

```

```

5 4 4.0
SubEnd
End

```

```

End Input
eor

```

```

mv RUNKF t21.NaCl

```

```

rm Points

```

The next run has largely the same input and provides a restart of the previous run.

The key RESTARTDOS tells the program to pick up the indicated file as restart file *and* to use it for DOS analysis purposes.

The DOS key block details the energy grid (and range) and the file to write the data to. The optional keys GROSSPOPULATIONS and OverlapPopulations invoke the computation of, respectively, gross populations and overlap populations (i.e. for each of these the density-of-states values in the user-defined energy grid).

```

$ADFBIN/band << eor
Title Title NaCl (from neutral atoms)  DOS analysis (restart)

```

```

Comment
Technical
  Hybrid K space integration (3D)
  Low real space integration accuracy
  Natural coordinates
  Lengths in Angstrom
  Parameters Dirac procedure

```

## Features

Lattice : 3D  
Unit cell : 2 atoms  
Basis : NO w/ core  
Options : Use restart file for DOS  
Analysis: DOS, PDOS, COOP

End

maxmemoryusage 20

Restart t21.NaCl &amp;

DOS

End

Accuracy 3.5

Kspace 3

SCF

Iterations 15

End

Units

Length Angstrom

End

Lattice

0 2.75 2.75

2.75 0 2.75

2.75 2.75 0

End

DOS

File NaCl.dos

Energies 1000

Min -0.5

Max 0.5

End

GROSSPOPULATIONS

FRAG 1

FRAG 2

SUM

1 0

2 0

ENDSUM

End

OVERLAPPOPULATIONS

LEFT

FRAG 1

RIGHT

FRAG 2

LEFT

1 0

1 1

RIGHT

2 0

2 1

End

Atoms NA

0

End

Coordinates Natural

Atoms CI

.5 .5 .5

End

AtomType Na

Dirac Na

4 1

Radial 2000

RMin 1E-4

RMax 60

VALENCE

1 0

2 0

2 1

3 0 1.0

SubEnd

FitFunctions

1 0 18.9

2 0 30.3

2 0 15.5

3 0 14.9

3 0 8.9

4 0 7.8

4 0 5.1

4 0 3.3

5 0 2.8

5 0 1.9

5 0 1.3

2 1 14.3

3 1 9.9

4 1 6.7

4 1 3.4

5 1 2.4

5 1 1.3

3 2 10.5

4 2 5.4

5 2 3.0

5 2 1.3

4 3 5.8

5 3 1.7

5 4 2.0

SubEnd

End

AtomType Cl

Dirac Cl

5 3

VALENCE

1 0

2 0

2 1

3 0

3 1 5.0

SubEnd

FitFunctions

1 0 29.1

2 0 49.5

2 0 26.1

3 0 25.8

3 0 15.8

4 0 14.2

4 0 9.4

4 0 6.2

5 0 5.4

5 0 3.8

5 0 2.6

2 1 21.2

3 1 16.5

4 1 12.4

4 1 6.8

5 1 5.1

5 1 3.1

3 2 16.6

4 2 9.4

5 2 5.5

5 2 2.6

4 3 8.7

5 3 3.3

5 4 4.0

SubEnd

End

End Input

eor

Finally, we copy the contents of the DOS result file to standard output

```
echo Contents of DOS file
```

```
cat NaCl.dos
```

## Cu\_slab: 2-dim. Periodic System

*Sample directory:* band/e\_Cu\_slab/

A two-dimensional infinite (periodic boundary conditions) slab calculation is performed for Cu. The dimensionality is simply defined by the number of records in the Lattice data block. In a 2-dimensional calculation the lattice vectors are put in the xy-plane. (In a one-dimensional calculation (polymer), the lattice vector is taken along the x-axis in the program.)

In this example, the minimal numerical basis from the DIRAC subprogram is augmented by Slater-type orbital basis functions, with the keyword BasisFunctions.

Slab calculations for metals frequently suffer from SCF convergence problems, as a result of the open valence band(s). To help the program converge it is often useful or even necessary to use some special features, such as the AllBands key. This particular key requires a numerical value (0.025 in the example) and implies that a finite-temperature electronic distribution is used, rather than a sharp cut-off at the Fermi level. The numerical value is the applied energy width, in hartree units.

The so-modified electronic distribution also affects the energy. The 'true' zero-T energy is computed, approximately, by an interpolation formula. The interpolation is not very accurate and one should try to use as small as possible values for the AllBands key so as to avoid increasing uncertainty in the results. The program prints, in the energy section of the output file, the finite-T correction term that has been applied through the interpolation formula. This gives at least an indication of any remaining uncorrected deviation of the outcome from a true zero-T calculation.

```
$ADFBIN/band << eor
```

```
Title Cu slab
```

```
Comment
```

```
Technical
```

```
  Quadratic K space integration
```

```
  Good real space integration accuracy
```

```
Features
```

```
  Lattice   : 2D
```

```
  Unit cell : 1 atom, 1x1
```

```
  Basis     : NO+STO w/ core
```

```
  Options   : AllBands (temperature effect)
```

```
End
```

```
MaxMemoryUsage=20
```

```
Kspace 5
```

```
Accuracy 4
```

```
Convergence
```

```
  AllBands 0.025
```

```
End
```

```
Lattice
```

```
  4.822 0.0
```

```
  0.0 4.822
```

```
End
```

```
Atoms Cu
  0.0  0.0  0.0
End
```

```
AtomType Cu
Dirac Cu
  7 5
VALENCE
  1 0 2.0
  2 0 2.0
  2 1 6.0
  3 0 2.0
  3 1 6.0
  3 2 10.0
  4 0 1.0
SubEnd
```

```
BasisFunctions
  3 2 1.65
  4 0 1.0
  4 1 2.0
SubEnd
```

```
FitFunctions
  1 0 44.50
  2 0 43.48
  3 0 38.92
  4 0 33.87
  4 0 23.32
  5 0 20.07
  5 0 14.33
  5 0 10.22
  6 0 8.77
  6 0 6.43
  6 0 4.72
  7 0 4.04
  7 0 3.03
  7 0 2.27
  2 1 34.50
  3 1 25.75
  4 1 19.17
  5 1 14.36
  5 1 8.97
  6 1 6.78
  6 1 4.40
  3 2 29.15
  4 2 17.85
  5 2 11.23
  5 2 5.94
  6 2 3.83
  6 2 2.13
  4 3 19.15
  4 3 8.05
  5 3 4.37
```

5 3 2.00  
5 4 13.80  
5 4 7.25  
5 4 3.81  
5 4 2.00

SubEnd  
End

EndInput  
eor

## CO absorption on a Cu slab

*Sample directory:* band/e\_Frags\_COcu/

This example illustrates the usage of fragments in a BAND calculation, for analysis purposes. The setup involves first the computation of the free CO overlayer, which is to be absorbed on a Cu surface. To suppress (most of the) interactions between the CO molecules, i.e. to effectively get the *molecular* CO, the KSpace parameter is set to 1 (= no dispersion), and the lattice parameters are set so large that the CO molecules are far apart.

The RUNKF key is used to save the standard result file, under the name 't21.CO'.

```
$ADFBIN/band << eor
```

```
O fragment
```

```
Comment
```

```
Technical
```

```
Zero order k space integration
```

```
Good real space integration accuracy
```

```
Definitions of variables
```

```
Features
```

```
Lattice : 2D, large lattice vectors
```

```
Unit cell : 2 atoms, 1x1, quasi molecular
```

```
Basis : NO+STO w/ core
```

```
Options : Save RUNKF restart (fragment) file
```

```
Prepare fragment for follow up
```

```
End
```

```
MAXMEMORYUSAGE 40
```

```
Save RUNKF ! save RUNKF as fragment file
```

```
Basis
```

```
PrepareFragment ! keep all bands, not only the occupied ones
```

```
End
```

```
PRINT EIGENS
```

```
Kspace 1 ! neglect dispersion
```

```
Accuracy 4
```

```
Define
```

```
bond=2.18
```

```
far=25
```

```
End
```

```
Lattice ! CO molecules far apart
```

```

    far 0.0
    0.0 far
End

```

```

Atoms C
  0 0 0
End

```

```

Atoms O
  0 0 bond
End

```

```

....
End Input
eor

```

```
mv RUNKF t21.CO
```

the fragment(s) to use: the file (t21.CO) and the numbering of atoms on the fragment file versus their occurrence in this calculation.

With FragLabels we assign names to the different symmetry orbitals.

The Density-of-States analysis details are given with the keys DOS (energy grid, result file with DOS data) and, optionally, GrossPopulations and OverlapPopulations.

```

$ADFBIN/band << eor
Title Cu slab with CO adsorbed

```

```

Comment
  Technical
    Quadratic K space integration (low)
    Good real space integration accuracy
    Definitions of variables
  Features
    Lattice   : 2D
    Unit cell : 3 atoms, 1x1
    Basis     : NO+STO w/ core
    Options   : Molecular fragment
              Analysis: DOS, PDOS, COOP
End

```

```
MaxMemoryUsage 40
```

```

Kspace 3
Accuracy 4

```

```
! fragment options
```

```

Basis
  SimpleFrag
End

```

```
NATOMSASFRAGMENT 2
```

Fragments t21.CO

1 1

2 2

End

Fragmentlabels

2Sigma

2Sigma\*

1Pi\_x

1Pi\_y

3Sigma

1Pi\_x\*

1Pi\_y\*

3Sigma\*

End

DOS ! Analysis

FILE pdos.CO\_Cu

ENERGIES 500

MIN -0.750

MAX 0.300

End

GROSSPOPULATIONS

3 2 ! All metal d states

SUM ! All metal sp states

3 0

3 1

ENDSUM

FRAG 1 ! All CO states

SUM ! CO 1pi

FRAGFUN 1 5

FRAGFUN 1 6

ENDSUM

FRAGFUN 1 7 ! CO 5-sigma

End

OVERLAPPOPULATIONS

LEFT ! Metal d with CO

3 2

RIGHT

FRAG 1

End

Define

dist=3.44

bond=2.18

End

Lattice

4.822 0.0

0.0 4.822

End

```
Atoms C
  0 0 dist
End

Atoms O
  0 0 dist+bond
End

Atoms CU
  0.0 0.0 0.0
End

...

End Input
eor
```

Finally, we copy the computed DOS data from the DOS result file to standard output.

```
echo Contents of DOS file
cat pdos.CO_Cu
```

## Polyacetylene polymer calculation

*Sample directory:* band/e\_CnHn/

This example illustrates how a one-dimensional periodic system can be treated by specifying only one lattice vector. It further shows how variables can be defined with the DEFINE keyword. The rest more or less speaks for itself. The Kspace integration is taken very accurate, whereas real space integration (ACCURACY keyword) is not so accurate.

Here and in the following BAND examples, we will leave out some space consuming parts of the input file which have been discussed already. Please check the actual input files if you wish to repeat one of the calculations.

```
$ADFBIN/band << eor
```

```
Title Polymer
```

```
Comment
```

```
Technical
```

```
  Quadratic k space integration (1D)
```

```
  Low real space integration accuracy
```

```
  Definitions of variables
```

```
Features
```

```
  Lattice   : 1D, polymer
```

```
  Unit cell : 4 atoms
```

```
  Basis    : NO+STO w/ core
```

```
End
```

```
Kspace 5
```

```
Accuracy 3
```

```
Units
```

```
  Length Angstrom
```

```
  Angle Radian
```

```
End
```

```
Define
```

```
  dCCd=1.3386
```

```
  dCCs=1.4510
```

```
  dCH=1.0770
```

```
  aCCC=124.5/180*pi
```

```
  arC=aCCC-pi/2
```

```
  aCCH=119.2/180*pi ! double bonded CC
```

```
  arH=aCCH-pi/2
```

```
End
```

```
Lattice
```

```
dCCd+sin(arC)*dCCs cos(arC)*dCCs 0.0
```

```
End
```

```
Atoms C
```

```
dCCd/2 0.0 0.0
-dCCd/2 0.0 0.0
End
```

```
Atoms H
dCCd/2+sin(arH)*dCH -cos(arH)*dCH 0.0
-dCCd/2-sin(arH)*dCH cos(arH)*dCH 0.0
End
```

A larger unit cell can of course be specified as well. In the second part of the example a supercell of 5 units is used. Another new feature introduced in this example is the TAILS keyword, which similar to ADF implies that distance cut-offs are applied to make the calculation cheaper. At the moment no big gains are yet to be expected from this, but this situation is expected to change in future versions of the code. Another keyword which is relevant for saving computer time in calculations on large systems is the NONORTHOGONALBASIS subkey

in the BASIS key. This subkey is actually mandatory at the moment if the TAILS keyword is used.

```
$ADFBIN/band << eor
Title Polymer with big unit cell (5 units)
```

```
Comment
Technical
  Low quadratic k space integration (1D)
  Low real space integration accuracy
  Definitions of variables
Features
  Lattice : 1D, polymer
  Unit cell : 4 atoms
  Basis : NO+STO w/ core
End
```

```
Kspace 3
Accuracy 3
```

```
Units
Length Angstrom
Angle Radian
End
```

```
Tails Bas=1E-2
```

```
Basis
NonOrthogonalSCFBasis
End
```

```
Define
dCCd=1.3386
dCCs=1.4510
dCH=1.0770
aCCC=124.5/180*pi
arC=aCCC-pi/2
aCCH=119.2/180*pi ! double bonded CC
arH=aCCH-pi/2
Latx(nlatt)=nlatt*(dCCd+sin(arC)*dCCs)
```

```
Laty(nlatt)=nlatt*(cos(arC)*dCCs)
Laty(nlatt)=nlatt*(cos(arC)*dCCs)
End
```

```
Lattice
Latx(5) Laty(5) 0.0
End
```

```
Atoms C
dCCd/2 0.0 0.0
-dCCd/2 0.0 0.0
dCCd/2+Latx(1) Laty(1) 0.0
-dCCd/2+Latx(1) Laty(1) 0.0
dCCd/2+Latx(2) Laty(2) 0.0
-dCCd/2+Latx(2) Laty(2) 0.0
dCCd/2+Latx(3) Laty(3) 0.0
-dCCd/2+Latx(3) Laty(3) 0.0
dCCd/2+Latx(4) Laty(4) 0.0
-dCCd/2+Latx(4) Laty(4) 0.0
End
```

```
Atoms H
dCCd/2+sin(arH)*dCH -cos(arH)*dCH 0.0
-dCCd/2-sin(arH)*dCH cos(arH)*dCH 0.0
dCCd/2+sin(arH)*dCH+Latx(1.0) -cos(arH)*dCH+Laty(1.0) 0.0
-dCCd/2-sin(arH)*dCH+Latx(1.0) cos(arH)*dCH+Laty(1.0) 0.0
dCCd/2+sin(arH)*dCH+Latx(2.0) -cos(arH)*dCH+Laty(2.0) 0.0
-dCCd/2-sin(arH)*dCH+Latx(2.0) cos(arH)*dCH+Laty(2.0) 0.0
dCCd/2+sin(arH)*dCH+Latx(3.0) -cos(arH)*dCH+Laty(3.0) 0.0
-dCCd/2-sin(arH)*dCH+Latx(3.0) cos(arH)*dCH+Laty(3.0) 0.0
dCCd/2+sin(arH)*dCH+Latx(4.0) -cos(arH)*dCH+Laty(4.0) 0.0
-dCCd/2-sin(arH)*dCH+Latx(4.0) cos(arH)*dCH+Laty(4.0) 0.0
End
```

## Relativistic effects: Platinum slab

Sample directory: band/e\_Pt\_slab/

This example can of course be compared directly to the Cu slab. This example is important, as SCF convergence is frequently difficult in slab calculations. The specifications in the CONVERGENCE, SCF, and DIIS blocks are typical. Such settings are recommended in slab calculations with convergence problems.

The DEGENERATE subkey specifies that bands with the same energy should have the same occupation numbers. This helps SCF convergence. The same is true for the values for the MIXING subkey in the SCF block and the DIMIX subkey in the DIIS block. Please note that the recommended value for Mixing is approximately half of the value for Dimix.

Another important feature in BAND is that it enables relativistic treatments for heavy nuclei. Both the ZORA scalar relativistic option and spin-orbit effects have been implemented. The line

Relativistic ZORA SPIN

specifies that in this case both the scalar relativistic effects (ZORA) and spin-orbit effects (SPIN) will be taken into account. Whereas the ZORA keyword does not make the calculation much more time-consuming, the same cannot be said for the spin-orbit option. Usually the ZORA keyword will give the most pronounced relativistic effects and the spin-orbit effects will be a fairly minor correction to that. We therefore recommend scalar ZORA as a good default method for treating heavy nuclei.

The DEPENDENCY keyword means that the calculation should continue even if the basis is nearly linearly dependent (as measured by the eigenvalues of the overlap matrix).

```
$ADFBIN/band << eor  
Title Platinum slab
```

Comment

Technical

Low quadratic K space integration  
Low real space integration accuracy

Features

Lattice : 2D  
Unit cell : 3 atoms, 1x1  
Basis : NO+STO w/ core  
Options : Spinorbit ZORA

End

Convergence

Degenerate 1.0E-03  
End

SCF

Iterations 60  
Mixing 0.06  
End

DIIS

NCycleDamp 15  
DiMix 0.15

End

KSpace 3  
Accuracy 3

Relativistic ZORA SPIN

Dependency Basis=1E-8

Define  
latt=7.41  
lvec=latt/SQRT(2.0)  
ysh=lvec/SQRT(3.0)  
dlay=latt/SQRT(3.0)  
End

Lattice  
SQRT(3.0)\*lvec/2.0 0.5\*lvec  
SQRT(3.0)\*lvec/2.0 -0.5\*lvec  
End

Atoms Pt  
0 0 0 :: layer 1  
-ysh 0.0 -dlay :: layer 2  
ysh 0.0 -2.0\*dlay :: layer 3  
End

...

END INPUT  
eor

## Pd bulk

Sample directory: band/e\_Pd/

This example does not add much to what has already been treated above (e.g. NaCl bulk example). The main things to note are the specification of a scalar relativistic ZORA calculation, and the fact that symmetry information will be printed on output.

```
$ADFBIN/band << eor  
Title Pd bulk
```

```
Comment
```

```
Technical
```

```
Hybrid k space integration (3D)
```

```
Reasonable real space integration accuracy
```

```
Definitions of variables
```

```
Features
```

```
Lattice : 3D
```

```
Unit cell : 1 atom
```

```
Basis : NO+STO w/ core
```

```
Options : spin restricted, scalar relativistic,  
numerical fit functions  
Full symmetry
```

```
End
```

```
KGRPX 1
```

```
Kspace 5
```

```
Accuracy 4.0
```

```
Print SYMMETRY
```

```
Relativistic ZORA
```

```
Define
```

```
halflatt=7.35/2
```

```
End
```

```
Lattice :: FCC
```

```
0 halflatt halflatt
```

```
halflatt 0 halflatt
```

```
halflatt halflatt 0
```

```
End
```

```
Atoms Pd
```

```
0.0
```

```
End
```

```
...
```

```
END INPUT
```

```
eor
```

## Hydrogen on Pt surface

Sample directory: band/e\_H\_on\_Pt/

This example is in many ways similar to the Pt slab example. We refer to the explanations in that example for details. Suffice it to say here that the convergence criteria have again been chosen such that also difficult slab calculations have a good chance to converge. Further, this calculation is once again at the spin-orbit relativistic level. The geometry is such that two layers of Pt are covered by a hydrogen layer.

```
$ADFBIN/band << eor
```

```
Title Hydrogen on platinum
```

```
Comment
```

```
Technical
```

```
  Low quadratic K space integration
```

```
  Low real space integration accuracy
```

```
Features
```

```
  Lattice : 2D
```

```
  Unit cell : 4 atoms, 1x1
```

```
  Basis : NO+STO
```

```
  Options : Spinorbit ZORA
```

```
End
```

```
SCF
```

```
  Mixing 0.1
```

```
  Iterations 100
```

```
End
```

```
Convergence
```

```
  Degenerate default
```

```
  Criterion 1e-6
```

```
End
```

```
DIIS
```

```
  NCycleDamp 0
```

```
  DiMix 0.15
```

```
End
```

```
KSpace 3
```

```
Accuracy 3
```

```
Relativistic ZORA SPIN
```

```
Dependency Basis=1E-8
```

```
Define
```

```
latt=7.41
```

```
lvec=latt/SQRT(2.0)
```

```
ysh=lvec/SQRT(3.0)
```

```
dlay=latt/SQRT(3.0)
```

```
height=3.0
```

```
End
```

```
Lattice
  SQRT(3.0)*Ivec/2.0 0.5*Ivec
  SQRT(3.0)*Ivec/2.0 -0.5*Ivec
End
```

```
Atoms Pt
  0 0 0 :: layer 1
  -ysh 0.0 -dlay :: layer 2
End
```

```
Atoms H
  0.0 0.0 height
End
...
```

```
END INPUT
eor
```

## NiCu surface alloy with GGA xc potential

Sample directory: band/e\_NiCu\_XC/

This is an important example featuring many important keywords.

The line:

Skip DOS

reduces the length of the output considerably.

A spin-unrestricted calculation can be done by using the keyword

Unrestricted

The Becke Perdew generalized gradient approximation (GGA) for the exchange-correlation functional is invoked through the XC block:

XC

GGA Always Becke Perdew

End

The subkey ALWAYS indicates that the Becke-Perdew xc potential should be used during the SCF. If this is not specified, the GGA energy will be calculated post-SCF using the LDA density.

\$ADFBIN/band << eor

Title Surface alloy: Cu slab with one surface Cu replaced by Ni (1:1)

Comment

Technical

Low quadratic K space integration

Reasonable real space integration accuracy

Features

Lattice : 2D

Unit cell : 4 atoms, sqrt(2) x sqrt(2), 2 layers

Basis : NO+STO w/ core

Options : XC functional in SCF

End

Skip DOS

Convergence

Degenerate default

End

DIIS

NCycleDamp 15

DiMix 0.25

End

SCF

Mixing 0.10

End

```
Accuracy 3.5
Kspace 3

Unrestricted

XC
  GGA Always Becke Perdew
End

Units
  Length Angstrom
End

Define
latt=3.61 ! FCC lattice parameter
halflatt=latt/2
End

Lattice
latt 0 0
0 latt 0
End

Atoms Ni
0 0 0
End

Atoms Cu
:: layer 1
halflatt halflatt 0
:: layer 2
0 halflatt -halflatt
halflatt 0 -halflatt
End

...
EndInput
eor
```

## Confinement,tails, and nonorthogonalSCFbasis options for copper dimer

Sample directory: band/e\_Cu2\_Confine/

This example illustrates three keywords to speed up BAND calculations and reduce the required disk space to some extent, without significant loss of accuracy. The nonorthogonalscfbasis option is likely to become the default in future versions. We already recommend it for standard use. It implies a different technical solution for the SCF equations that should not alter the results, but which should typically be faster.

The tails keyword implies that cutoffs will be applied to the (basis) functions. Basis functions will be assumed to be zero outside the radius where the remaining relative norm of the function is smaller than 0.00001, in this example. The shape of the function is slightly modified, in a "soft" way, beyond the radius where the relative norm of the function outside the radius is 0.01. The Coulomb option is always recommended in this line as it gives more accurate results for core functions. The tails option works only if the nonorthogonalscfbasis option is specified.

The confinement option can be specified for each atom type separately. It is based on a soft cut-off related to a distance criterion. In this sense it is quite different from the tails option. The confinement option can be useful in itself, for avoiding near linear dependency in the basis. However, its use in making BAND calculations more efficient is primarily in combination with the tails option.

These options became available in ADF2002.03 for the first time.

```
$ADFBIN/band << eor
```

```
Title Copper dimer - confinement and tails options
```

```
KSpace 1
```

```
Accuracy 4.5
```

```
Basis  
nonorthogonalscfbasis  
end
```

```
Dependency Basis=1E-10 Fit=1E-7
```

```
Units  
Angle Radian  
End
```

```
SCF  
Mixing 0.05  
Iterations 20  
End
```

```
Convergence  
Degenerate default  
End
```

```
DIIS  
NCycleDamp 0
```

```
DiMix 0.15  
End
```

```
Define  
bbb=20  
ccc=4.2  
End
```

```
Lattice  
  bbb bbb 0  
 -bbb bbb 0  
End
```

```
Atoms Cu  
  0 0 0  
  0 0 ccc  
End
```

```
....  
END INPUT  
eor
```

The output file prints some information on how much was gained because of the tails and confinement options in various routines, for example:

Compression due to negligible tails in baspnt

max. percentage stored per node 98.23%

min. percentage stored per node 95.45%

## Time-dependent DFT calculations for bulk silicon

Sample directory: band/e\_Silicon/

The time-dependent DFT functionality is important new functionality in ADF2002.03. It enables you to calculate frequency-dependent dielectric functions for 1-dimensional and 3-dimensional periodic systems. In the present example, a standard geometry for bulk Silicon is given. The accuracy and kspace variables can keep their normal values. The important part in this example is of course the RESPONSE block. It specifies that 7 frequencies should be treated, with an even spacing between 0.0 hartree and 0.25 hartree. In this example scalar ZORA relativistic effects are switched on with the isz line in the RESPONSE block.

```
$ADFBIN/band << eor
TITLE Silicon
```

```
ACCURACY 5
KSPACE 2
```

```
DEPENDENCY BASIS 1e-10
```

```
UNITS
LENGTH ANGSTROM
END
```

```
RESPONSE
nfreq 7
strtfr 0d0
endfr 25d-2
isz 1
END
```

```
DEFINE
AAA=5.43
HA=AAA/2
END
```

```
LATTICE
0 HA HA
HA 0 HA
HA HA 0
END
```

```
ATOMS Si
0.0 0.0 0.0
HA/2 HA/2 HA/2
END
```

```
...
END INPUT
eor
```

For Silicon the real and imaginary parts of the dielectric function:  $\epsilon(\omega) = 1 + 4 \text{ Pi } \chi(\omega)$  are

calculated.

In the output file, the results will look something like the fragment below. The output specifies for which frequency the dielectric function is determined, and then proceeds to print the values for the 3x3 tensors.

The real and imaginary parts are printed separately. At this frequency, the imaginary part is still zero. Because of the high symmetry of the system, the real part is a constant times the unit matrix except for numerical noise.

```

** Frequency ** 0.833333E-01 au 2.26756 eV **
** Start the SCF procedure **
**** Real ****
** Chi_ij X ** -12.8363 0.142802E-18 0.547977E-17
** Chi_ij Y ** 0.202883E-17 -12.8363 0.121052E-17
** Chi_ij Z ** 0.124042E-16 0.215311E-17 -12.8363
**** Imag ****
** Chi_ij X ** 0.000000E+00 0.000000E+00 0.000000E+00
** Chi_ij Y ** 0.000000E+00 0.000000E+00 0.000000E+00
** Chi_ij Z ** 0.000000E+00 0.000000E+00 0.000000E+00
*****

```

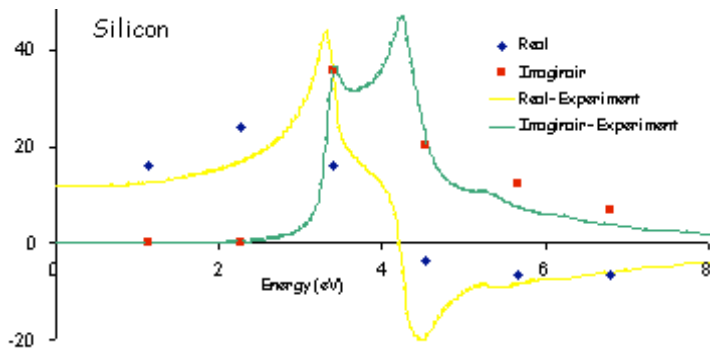
After each frequency has been treated, the results are summarized in a table, for each component separately. For the xx-component, this looks as in the table below. The frequency/energy is again printed in two different units, the Dielectric Function is printed in a.u. The values for Chi, which are trivially related to those printed here, are summarized in a separate table.

```

=====
==      Frequency      ==      Dielectric Function      ==
==   a.u. == e.V. ==      Re ==      Im ==
=====XX-dir=====
0.416667E-01 1.13378 16.1119 0.000000E+00
0.833333E-01 2.26756 23.7904 0.000000E+00
0.125000 3.40134 15.8529 35.8574
0.166667 4.53512 -3.49949 20.2221
0.208333 5.66890 -6.60897 12.3661
0.250000 6.80268 -6.42943 6.87957
=====YY-dir=====
0.416667E-01 1.13378 16.1119 0.000000E+00
0.833333E-01 2.26756 23.7904 0.000000E+00
0.125000 3.40134 15.8529 35.8574
0.166667 4.53512 -3.49949 20.2221
0.208333 5.66890 -6.60897 12.3661
0.250000 6.80268 -6.42943 6.87957
=====ZZ-dir=====
0.416667E-01 1.13378 16.1119 0.000000E+00
0.833333E-01 2.26756 23.7904 0.000000E+00
0.125000 3.40134 15.8529 35.8574
0.166667 4.53512 -3.49949 20.2221
0.208333 5.66890 -6.60897 12.3661
0.250000 6.80268 -6.42943 6.87957

```

Results of the test calculation (red/blue) are plotted in next Figure together with experimental data (yellow/green). The results for the seven specified frequencies are given. It should be obvious that more frequencies are needed (resulting in longer run times) to obtain a smooth curve in which peaks cannot be missed because of too coarse interpolation.



## Time-dependent DFT calculations for hydrogen chain

Sample directory: band/e\_H\_chain

The input for a one-dimensional system is no different from that for a three-dimensional system. No tests have been performed on two-dimensional systems yet in combination with the TDDFT code. It is therefore not expected to work. Besides the number of frequencies, and the beginning and end frequency of the frequency range, the RESPONSE block also contains stricter than default convergence criteria (cnvi and cnvj) for the fit coefficients describing the density change.

```
$ADFBIN/band << eor
```

```
Title H2-chain
```

```
ACCURACY 5
```

```
KSPACE 3
```

```
DEPENDENCY BASIS 1e-10
```

```
RESPONSE
```

```
nfreq 10
```

```
strtfr 0d0
```

```
endfr 15d-3
```

```
cnvi 1d-5
```

```
cnvj 1d-5
```

```
END
```

```
DEFINE
```

```
HX = 4.5
```

```
END
```

```
LATTICE
```

```
HX
```

```
END
```

```
ATOMS H
```

```
1.0 0.001 0.0
```

```
-1.0 -0.001 0.0
```

```
END
```

```
...
```

```
END INPUT
```

```
eor
```

The output for this calculation will give rise to a table like the following one:

```
=====
==      Frequency      ===      Polarizability(xx)  ==
==      a.u. ==      e.V. ===      Re ==      Im ==
=====
      0.166667E-02  0.453512E-01  127.119  0.00000
      0.333333E-02  0.907023E-01  127.201  0.00000
```

0.500000E-02	0.136054	127.338	0.00000		
0.666667E-02	0.181405	127.530	0.00000		
0.833333E-02	0.226756	127.778	0.00000		
0.100000E-01	0.272107	128.083	0.00000		
0.116667E-01	0.317458	128.446	0.00000		
0.133333E-01	0.362809	128.868	0.00000		
0.150000E-01	0.408161	129.351	0.00000		
=====					
==	Frequency	===	Chi_JJ (xx)	==	
==	a.u.	== e.V.	=== Re	== Im	==
=====					
0.00000	0.00000	-2.74118	0.00000		
0.166667E-02	0.453512E-01	-2.74153	0.00000		
0.333333E-02	0.907023E-01	-2.74259	0.00000		
0.500000E-02	0.136054	-2.74436	0.00000		
0.666667E-02	0.181405	-2.74685	0.00000		
0.833333E-02	0.226756	-2.75005	0.00000		
0.100000E-01	0.272107	-2.75399	0.00000		
0.116667E-01	0.317458	-2.75866	0.00000		
0.133333E-01	0.362809	-2.76409	0.00000		
0.150000E-01	0.408161	-2.77028	0.00000		
=====					